Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques

APPROX/RANDOM 2020, August 17–19, 2020,
Virtual Conference

Edited by

Jarosław Byrka

Raghu Meka



LIPIcs – Vol. 176 – APPROX/RANDOM 2020 www.dagstuhl.de/lipics

Editors

Jarosław Byrka
University of Wrocław, Poland
jby@cs.uni.wroc.pl

Raghu Meka
University of California, Los Angeles, USA
raghuvardhan@gmail.com

ACM Classification 2012
Mathematics of computing; Theory of computation

ISBN 978-3-95977-164-1

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/978-3-95977-164-1.

Publication date
August, 2020

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 3.0 Unported license (CC-BY 3.0): https://creativecommons.org/licenses/by/3.0/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.2020.0

LIPIcs – Leibniz International Proceedings in Informatics

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

Editorial Board

- Luca Aceto (*Chair*, Gran Sasso Science Institute and Reykjavik University)
- Christel Baier (TU Dresden)
- Mikolaj Bojanczyk (University of Warsaw)
- Roberto Di Cosmo (INRIA and University Paris Diderot)
- Javier Esparza (TU München)
- Meena Mahajan (Institute of Mathematical Sciences)
- Dieter van Melkebeek (University of Wisconsin-Madison)
- Anca Muscholl (University Bordeaux)
- Luke Ong (University of Oxford)
- Catuscia Palamidessi (INRIA)
- Thomas Schwentick (TU Dortmund)
- Raimund Seidel (Saarland University and Schloss Dagstuhl – Leibniz-Zentrum für Informatik)

ISSN 1868-8969

https://www.dagstuhl.de/lipics

Contents

Preface
Jarosław Byrka and Raghu Meka 0:ix

RANDOM

Extractor Lower Bounds, Revisited
Divesh Aggarwal, Siyao Guo, Maciej Obremski, João Ribeiro, and Noah Stephens-Davidowitz 1:1–1:20

A Simpler Strong Refutation of Random k-XOR
Kwangjun Ahn 2:1–2:15

Iterated Decomposition of Biased Permutations via New Bounds on the Spectral Gap of Markov Chains
Sarah Miracle, Amanda Pascoe Streib, and Noah Streib 3:1–3:21

Improved Explicit Hitting-Sets for ROABPs
Zeyu Guo and Rohit Gurjar 4:1–4:16

Almost Optimal Testers for Concise Representations
Nader H. Bshouty 5:1–5:20

Palette Sparsification Beyond $(\Delta + 1)$ Vertex Coloring
Noga Alon and Sepehr Assadi 6:1–6:22

On Hitting-Set Generators for Polynomials That Vanish Rarely
Dean Doron, Amnon Ta-Shma, and Roei Tell 7:1–7:23

Polynomial Identity Testing for Low Degree Polynomials with Optimal Randomness
Markus Bläser and Anurag Pandey 8:1–8:13

Bounds for List-Decoding and List-Recovery of Random Linear Codes
Venkatesan Guruswami, Ray Li, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters 9:1–9:21

Is It Possible to Improve Yao's XOR Lemma Using Reductions That Exploit the Efficiency of Their Oracle?
Ronen Shaltiel 10:1–10:20

Balanced Allocation on Dynamic Hypergraphs
Catherine Greenhill, Bernard Mans, and Ali Pourmiri 11:1–11:22

The GaussianSketch for Almost Relative Error Kernel Distance
Jeff M. Phillips and Wai Ming Tai 12:1–12:20

A Fast Binary Splitting Approach to Non-Adaptive Group Testing
Eric Price and Jonathan Scarlett 13:1–13:20

Maximum Shallow Clique Minors in Preferential Attachment Graphs Have Polylogarithmic Size
Jan Dreier, Philipp Kuinke, and Peter Rossmanith 14:1–14:13

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

On Nonadaptive Security Reductions of Hitting Set Generators
Shuichi Hirahara and Osamu Watanabe 15:1–15:14

Testable Properties in General Graphs and Random Order Streaming
Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler 16:1–16:20

Multicriteria Cuts and Size-Constrained k-Cuts in Hypergraphs
Calvin Beideman, Karthekeyan Chandrasekaran, and Chao Xu 17:1–17:21

On Testing and Robust Characterizations of Convexity
Eric Blais and Abhinav Bommireddi 18:1–18:15

Distributed Testing of Graph Isomorphism in the CONGEST Model
Reut Levi and Moti Medina 19:1–19:24

Reaching a Consensus on Random Networks: The Power of Few
Linh Tran and Van Vu 20:1–20:15

Time-Space Tradeoffs for Distinguishing Distributions and Applications to Security of Goldreich's PRG
Sumegha Garg, Pravesh K. Kothari, and Ran Raz 21:1–21:18

Streaming Verification for Graph Problems: Optimal Tradeoffs and Nonlinear Sketches
Amit Chakrabarti, Prantar Ghosh, and Justin Thaler 22:1–22:23

Disjointness Through the Lens of Vapnik–Chervonenkis Dimension: Sparsity and Beyond
Anup Bhattacharya, Sourav Chakraborty, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar 23:1–23:15

Testing Data Binnings
Clément L. Canonne and Karl Wimmer 24:1–24:13

Chernoff Bound for High-Dimensional Expanders
Tali Kaufman and Ella Sharakanski 25:1–25:22

Vector-Matrix-Vector Queries for Solving Linear Algebra, Statistics, and Graph Problems
Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu 26:1–26:20

Almost Optimal Distribution-Free Sample-Based Testing of k-Modality
Dana Ron and Asaf Rosin 27:1–27:19

When Is Amplification Necessary for Composition in Randomized Query Complexity?
Shalev Ben-David, Mika Göös, Robin Kothari, and Thomas Watson 28:1–28:16

On Multilinear Forms: Bias, Correlation, and Tensor Rank
Abhishek Bhrushundi, Prahladh Harsha, Pooya Hatami, Swastik Kopparty, and Mrinal Kumar 29:1–29:23

On the List Recoverability of Randomly Punctured Codes
Ben Lund and Aditya Potukuchi 30:1–30:11

APPROX

On Perturbation Resilience of Non-Uniform k-Center
Sayan Bandyapadhyay 31:1–31:22

Low-Rank Binary Matrix Approximation in Column-Sum Norm
Fedor V. Fomin, Petr A. Golovach, Fahad Panolan, and Kirill Simonov 32:1–32:18

Pinning down the Strong Wilber 1 Bound for Binary Search Trees
Parinya Chalermsook, Julia Chuzhoy, and Thatchaphol Saranurak 33:1–33:21

Revisiting Alphabet Reduction in Dinur's PCP
Venkatesan Guruswami, Jakub Opršal, and Sai Sandeep 34:1–34:14

L_p Pattern Matching in a Stream
Tatiana Starikovskaya, Michal Svagerka, and Przemysław Uznański 35:1–35:23

Computing Bi-Lipschitz Outlier Embeddings into the Line
Karine Chubarian and Anastasios Sidiropoulos 36:1–36:21

Online Minimum Cost Matching with Recourse on the Line
Nicole Megow and Lukas Nölke 37:1–37:16

Hardness of Approximation of (Multi-)LCS over Small Alphabet
Amey Bhangale, Diptarka Chakraborty, and Rajendra Kumar 38:1–38:16

On Approximating Degree-Bounded Network Design Problems
Xiangyu Guo, Guy Kortsarz, Bundit Laekhanukit, Shi Li, Daniel Vaz, and Jiayi Xian 39:1–39:21

Permutation Strikes Back: The Power of Recourse in Online Metric Matching
Varun Gupta, Ravishankar Krishnaswamy, and Sai Sandeep 40:1–40:20

How to Cut a Ball Without Separating: Improved Approximations for Length Bounded Cut
Eden Chlamtáč and Petr Kolman 41:1–41:17

On the Facility Location Problem in Online and Dynamic Models
Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian 42:1–42:23

Nearly Optimal Embeddings of Flat Tori
Ishan Agarwal, Oded Regev, and Yi Tang 43:1–43:14

A Tight $(3/2 + \varepsilon)$ Approximation for Skewed Strip Packing
Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau 44:1–44:18

Learning Lines with Ordinal Constraints
Bohan Fan, Diego Ihara, Neshat Mohammadi, Francesco Sgherzi, Anastasios Sidiropoulos, and Mina Valizadeh 45:1–45:15

Improved Circular k-Mismatch Sketches
Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, Ely Porat, and Przemysław Uznański 46:1–46:24

On Guillotine Separability of Squares and Rectangles
Arindam Khan and Madhusudhan Reddy Pittu 47:1–47:22

Maximizing Throughput in Flow Shop Real-Time Scheduling
Lior Ben Yamin, Jing Li, Kanthi Sarpatwar, Baruch Schieber, and Hadas Shachnai 48:1–48:18

Maximizing the Correlation: Extending Grothendieck's Inequality to Large Domains
Dor Katzelnick and Roy Schwartz 49:1–49:18

Streaming Complexity of SVMs
Alexandr Andoni, Collin Burns, Yi Li, Sepideh Mahabadi, and David P. Woodruff 50:1–50:22

On the Parameterized Approximability of Contraction to Classes of Chordal Graphs
Spoorthy Gunda, Pallavi Jain, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale 51:1–51:19

Online Coloring of Short Intervals
Joanna Chybowska-Sokół, Grzegorz Gutowski, Konstanty Junosza-Szaniawski, Patryk Mikos, and Adam Polak 52:1–52:18

Approximating Requirement Cut via a Configuration LP
Roy Schwartz and Yotam Sharoni 53:1–53:16

Parametrized Metrical Task Systems
Sébastien Bubeck and Yuval Rabani 54:1–54:14

A Constant Factor Approximation for Capacitated Min-Max Tree Cover
Syamantak Das, Lavina Jain, and Nikhil Kumar 55:1–55:13

An Extension of Plücker Relations with Applications to Subdeterminant Maximization
Nima Anari and Thuy-Duong Vuong 56:1–56:16

Approximating Star Cover Problems
Buddhima Gamlath and Vadim Grinberg 57:1–57:19

On the Approximability of Presidential Type Predicates
Neng Huang and Aaron Potechin 58:1–58:20

An Approximation Algorithm for the MAX-2-Local Hamiltonian Problem
Sean Hallgren, Eunou Lee, and Ojas Parekh 59:1–59:18

Better and Simpler Learning-Augmented Online Caching
Alexander Wei 60:1–60:17

A 4/3-Approximation Algorithm for the Minimum 2-Edge Connected Multisubgraph Problem in the Half-Integral Case
Sylvia Boyd, Joseph Cheriyan, Robert Cummings, Logan Grout, Sharat Ibrahimpur, Zoltán Szigeti, and Lu Wang 61:1–61:12

Improved Multi-Pass Streaming Algorithms for Submodular Maximization with Matroid Constraints
Chien-Chung Huang, Theophile Thiery, and Justin Ward 62:1–62:19

Polylogarithmic Approximation Algorithm for k-Connected Directed Steiner Tree on Quasi-Bipartite Graphs
Chun-Hsiang Chan, Bundit Laekhanukit, Hao-Ting Wei, and Yuhao Zhang 63:1–63:20

Weighted Maximum Independent Set of Geometric Objects in Turnstile Streams
Ainesh Bakshi, Nadiia Chepurko, and David P. Woodruff 64:1–64:22

Preface

This volume contains the papers presented at the 23rd International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX 2020) and the 24th International Conference on Randomization and Computation (RANDOM 2020), which due to the travel restrictions related to COVID-19 were organized as parallel virtual conferences during August 17–19, 2020.

APPROX focuses on algorithmic and complexity issues surrounding the development of efficient approximate solutions to computationally difficult problems, and was the 23rd in the series. RANDOM is concerned with applications of randomness to computational and combinatorial problems, and was the 24th in the series. Prior to 2003, APPROX took place in Aalborg (1998), Berkeley (1999), Saarbrücken (2000), Berkeley (2001), and Rome (2002), while RANDOM took place in Bologna (1997), Barcelona (1998), Berkeley (1999), Geneva (2000), Berkeley (2001), and Harvard (2002). Since 2003, APPROX and RANDOM have been collocated, 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), and Boston (2019).

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

The volume contains 34 contributed papers, selected by the APPROX Program Committee out of 67 submissions, and 30 contributed papers, selected by the RANDOM Program Committee also out of 67 submissions. We would like to thank all of 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: Klaus Jansen, Samir Khuller, Monaldo Mastrolili, and László Végh for APPROX, and Oded Goldreich, Cris Moore, Anup Rao, Omer Reingold, Dana Ron, Ronitt Rubinfeld, Amit Sahai, Ronen Shaltiel, Alistair Sinclair, and Paul Spirakis for RANDOM.

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka

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

Program Committees

APPROX

Nikhil Bansal	CWI & TU Eindhoven
Jarosław Byrka (chair)	University of Wrocław
Andreas Emil Feldmann	Charles University in Prague
Naveen Garg	IIT Delhi
Anupam Gupta	Carnegie Mellon
Pasin Manurangsi	Google Research
Evangelos Markakis	AUEB, Athens
Nicole Megow	University of Bremen
Marcin Mucha	University of Warsaw
Harald Räcke	TU Munich
Laura Sanità	TU Eindhoven & University of Waterloo
Chaitanya Swamy	University of Waterloo
Jakub Tarnawski	Microsoft Research
Anke van Zuylen	William and Mary
David Williamson	Cornell University

RANDOM

Nima Anari	Stanford University
Eshan Chattopadhyay	Cornell University
Gil Cohen	Tel Aviv University
Parikshit Gopalan	VmWare Research
Prahladh Harsha	Tata Institute of Fundamental Research
Sam Hopkins	University of California, Berkeley
Valentine Kabanets	Simon Fraser University
Gautam Kamath	University of Waterloo
Tali Kaufman	Bar-Ilan University
Yin-Tat Lee	University of Washington
Sepideh Mahabadi	TTI Chicago
Raghu Meka	UCLA, PC chair
Jelani Nelson	University of California, Berkeley
Ryan O'Donnell	Carnegie Mellon University
Ilya Razenshteyn	Microsoft Research
Barna Saha	University of California, Berkeley
Tselil Schramm	Stanford University
Madhu Sudan	Harvard University
Avishay Tal	University of California, Berkeley
Eric Vigoda	Georgia Tech University
Mary Wootters	Stanford University

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Subreviewers

Amir Abboud
Jayadev Acharya
Marek Adamczyk
Vedat Alev Levi
Maryam Aliakbarpour
Alexandr Andoni
Tung Anh Vu
Per Austrin
Arturs Backurs
Soheil Behnezhad
Marcin Bienkowski
Alexander Birx
Andrej Bogdanov
Cornelius Brand
Matthew Brennan
Jonah Brown-Cohen
Mark Bun
Sarah Cannon
Clément Canonne
Marco Carmosino
Deeparnab Chakrabarty
Diptarka Chakraborty
Lin Chen
Jan Chorowski
Christian Coester
Roni Con
Graham Cormode
Ilias Diakonikolas
Michael Dinitz
Irit Dinur
Yann Disser
Dean Doron
Szymon Dudycz
Lech Duraj
Matthias Englert
Thomas Erlebach
Alireza Farhadi
Arnold Filtser
Zachary Friggstad
Kshitij Gajjar
Venkata Gandikota
Shay Golan
Alexander Golovnev
Jesse Goodman
Themistoklis Gouleakis
Rohit Gurjar
Tom Hayes
Chien-Chung Huang
Johan Håstad
Sharat Ibrahimpur
Huseyin Inan
Davis Issac
Naonori Kakimura
Sagar Kale
Michael Kapralov
Nikolai Karpov
Nathan Klein
Robert Kleinberg
Adam Klivans
Stavros Kolliopoulos
Weihao Kong
Christian Konrad
Pravesh Kothari
Jakub Kozik
Ravishankar Krishnaswamy
Janardhan Kulkarni
Akash Kumar
Pankaj Kumar
Ravi Kumar
Amit Kumar
Adam Kurpisz
Euiwoong Lee
Amit Levi
Mateusz Lewandowski
Xin Li
Jerry Li
Jason Li
Ray Li
Jyun-Jie Liao
Kuikui Liu
Daogao Liu
Hsiang-Hsuan Liu
Yang Liu
Anand Louis
Shachar Lovett
Yury Makarychev
Jan Marcinkowski
Andrew McGregor
Slobodan Mitrovic
Sidhanth Monanty

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Sagnik Mukhopadhyay
Cameron Musco
Tobias Mömke
Viswanath Nagarajan
Vasileios Nakos
Assaf Naor
Amir Nayyeri
Joe Neeman
Aleksandar Nikolov
Jakob Nordstrom
Orr Paradise
Alice Paul
Shir Peleg
Will Perkins
Seth Pettie
Michał Pilipczuk
Adam Polak
Aditya Potukuchi
Kirk Pruhs
Kent Quanrud
Miklos Racz Z.
Ashutosh Rai
R Ravi
Dana Ron
Noga Ron-Zewi
Chandan Saha
David Saulpic
Kevin Schewior
Paweł Schmidt
C. Seshadhri
Shashwat Silas
Paris Siminelakis
Bertrand Simon
Aleksa Stankovic
Thomas Steinke
Amnon Tashma
Roei Tell
Anamay Tengse
Przemysław Uznański
Rob van Stee
Pavel Veselý
Aravindan Vijayaraghavan
Ben Volk Lee
Tal Wagner
Erik Waingarten
Armin Weiss
Peter Winkler
Karol Węgrzycki
Huacheng Yu
Christos Zaroliagis
Samson Zhou
Anna Zych

List of Authors

Ishan Agarwal (43)
Courant Institute of Mathematical Sciences, New York University, NY, USA

Divesh Aggarwal (1)
National University of Singapore, Singapore

Kwangjun Ahn (2)
Department of EECS, Massachusetts Institute of Technology, Cambridge, MA, USA

Noga Alon (6)
Department of Mathematics, Princeton University, NJ, USA; Schools of Mathematics and Computer Science, Tel Aviv University, Israel

Afrouz Jabal Ameli (44)
IDSIA, USI-SUPSI, Manno, Switzerland

Nima Anari (56)
Department of Computer Science, Stanford University, CA, USA

Alexandr Andoni (50)
Columbia University, New York, NY, USA

Sepehr Assadi (6)
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Ainesh Bakshi (64)
Carnegie Mellon University, Pittsburgh, PA, USA

Sayan Bandyapadhyay (31)
Department of Informatics, University of Bergen, Norway

Calvin Beideman (17)
University of Illinois, Urbana-Champaign, IL, USA

Lior Ben Yamin (48)
Computer Science Department, Technion, Haifa, Israel

Shalev Ben-David (28)
University of Waterloo, Canada

Amey Bhangale (38)
University of California Riverside, CA, USA

Anup Bhattacharya (23)
Indian Statistical Institute, Kolkata, India

Abhishek Bhrushundi (29)
Rutgers University, Piscataway, NJ, USA

Eric Blais (18)
University of Waterloo, Canada

Markus Bläser (8)
Department of Computer Science, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany

Abhinav Bommireddi (18)
University of Waterloo, Canada

Sylvia Boyd (61)
School of Electrical Engineering and Computer Science, University of Ottawa, Canada

Nader H. Bshouty (5)
Department of Computer Science, Technion, Haifa, Israel

Sébastien Bubeck (54)
Microsoft Research, Redmond, WA, USA

Collin Burns (50)
Columbia University, New York, NY, USA

Clément L. Canonne (24)
IBM Research, Almaden, CA, USA

Amit Chakrabarti (22)
Dartmouth College, Hanover, NH, USA

Diptarka Chakraborty (38)
National University of Singapore, Singapore

Sourav Chakraborty (23)
Indian Statistical Institute, Kolkata, India

Parinya Chalermsook (33)
Aalto University, Finland

Chun-Hsiang Chan (63)
Department of Computer Science, University of Michigan, Ann Arbor, MI, USA

Karthekeyan Chandrasekaran (17)
University of Illinois, Urbana-Champaign, IL, USA

Nadiia Chepurko (64)
MIT, Cambridge, MA, USA

Joseph Cheriyan (61)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Eden Chlamtáč (41)
Ben Gurion University of the Negev, Beer Sheva, Israel

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka
Leibniz International Proceedings in Informatics
LIPICS Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

Karine Chubarian (36)
Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, IL, USA

Julia Chuzhoy (33)
Toyota Technological Institute at Chicago, IL, USA

Joanna Chybowska-Sokół (52)
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland

Robert Cummings (61)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Artur Czumaj (16)
Department of Computer Science and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Coventry, UK

Syamantak Das (55)
IIIT Delhi, India

Dean Doron (7)
Department of Computer Science, Stanford University, CA, USA

Jan Dreier (14)
Department of Computer Science, RWTH Aachen University, Germany

Bohan Fan (45)
Department of Computer Science, University of Illinois at Chicago, IL, USA

Hendrik Fichtenberger (16)
Department of Computer Science, TU Dortmund, Germany

Fedor V. Fomin (32)
Department of Informatics, University of Bergen, Norway

Buddhima Gamlath (57)
École Polytechnique Fédérale de Lausanne, Switzerland

Sumegha Garg (21)
Department of Computer Science, Princeton University, NJ, USA

Arijit Ghosh (23)
Indian Statistical Institute, Kolkata, India

Prantar Ghosh (22)
Dartmouth College, Hanover, NH, USA

Shay Golan (46)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Petr A. Golovach (32)
Department of Informatics, University of Bergen, Norway

Fabrizio Grandoni (44)
IDSIA, USI-SUPSI, Manno, Switzerland

Catherine Greenhill (11)
UNSW Sydney, Australia

Vadim Grinberg (57)
Toyota Technological Institute at Chicago, Chicago, IL, USA

Logan Grout (61)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Spoorthy Gunda (51)
Simon Fraser University, Burnaby, Canada

Siyao Guo (1)
New York University Shanghai, China

Xiangyu Guo (39, 42)
Department of Computer Science and Engineering, University at Buffalo, NY, USA

Zeyu Guo (4)
Department of Computer Science, University of Haifa, Israel

Varun Gupta (40)
University of Chicago, IL, USA

Rohit Gurjar (4)
Department of Computer Science and Engineering, IIT Bombay, India

Venkatesan Guruswami (9, 34)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Grzegorz Gutowski (52)
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Waldo Gálvez (44)
Technical University of Munich, Germany

Mika Göös (28)
Stanford University, CA, USA

Sean Hallgren (59)
Pennsylvania State University, State College, University Park, PA, USA

Prahladh Harsha (29)
Tata Institute of Fundamental Research, Mumbai, India

Pooya Hatami (29)
Dept. of Computer Science & Engineering, The Ohio State University, Columbus, OH, USA

Shuichi Hirahara (15)
National Institute of Informatics, Tokyo, Japan

Chien-Chung Huang (62)
CNRS, DI ENS, Université PSL, Paris, France

Neng Huang (58)
University of Chicago, IL, USA

Sharat Ibrahimpur (61)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Diego Ihara (45)
Department of Computer Science, University of Illinois at Chicago, IL, USA

Lavina Jain (55)
IIIT Delhi, India

Pallavi Jain (51)
Indian Institute of Technology Jodhpur, India

Klaus Jansen (44)
University of Kiel, Germany

Konstanty Junosza-Szaniawski (52)
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland

Dor Katzelnick (49)
Department of Computer Science, Technion, Haifa, Israel

Tali Kaufman (25)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Arindam Khan (44, 47)
Indian Institute of Science, Bangalore, India

Tomasz Kociumaka (46)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Petr Kolman (41)
Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic

Tsvi Kopelowitz (46)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Swastik Kopparty (29)
Dept. of Computer Science & Dept. of Mathematics, Rutgers University, Piscataway, NJ, USA

Guy Kortsarz (39)
Department of Computer Science, Rutgers University Camden, NJ, USA

Pravesh K. Kothari (21)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Robin Kothari (28)
Microsoft Quantum and Microsoft Research, Redmond, WA, USA

Ravishankar Krishnaswamy (40)
Microsoft Research India, Bangalore, India

Philipp Kuinke (14)
Department of Computer Science, RWTH Aachen University, Germany

Janardhan Kulkarni (42)
The Algorithms Group, Microsoft Research, Redmond, WA, USA

Mrinal Kumar (29)
Dept. of Computer Science & Engineering, IIT Bombay, India

Nikhil Kumar (55)
IIT Delhi, India

Rajendra Kumar (38)
IIT Kanpur, India; National University of Singapore, Singapore

Bundit Laekhanukit (39, 63)
ITCS, Shanghai University of Finance and Economics, China

Eunou Lee (59)
Pennsylvania State University, State College, University Park, PA, USA

Reut Levi (19)
Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya, Israel

Jing Li (48)
Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA

Ray Li (9)
Department of Computer Science, Stanford University, CA, USA

Shi Li (39, 42)
Department of Computer Science and Engineering, University at Buffalo, NY, USA

Yi Li (50)
Nanyang Technological University, Singapore, Singapore

Daniel Lokshtanov (51)
University of California, Santa Barbara, CA, USA

Ben Lund (30)
Department of Mathematics, Princeton University, NJ, USA

Sepideh Mahabadi (50)
Toyota Technological Institute at Chicago, IL, USA

Bernard Mans (11)
Macquarie University, Sydney, Australia

Moti Medina (19)
School of Electrical & Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva, Israel

Nicole Megow (37)
Department for Mathematics and Computer Science, University of Bremen, Germany

Patryk Mikos (52)
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Sarah Miracle (3)
University of St. Thomas, St. Paul, MN, USA

Gopinath Mishra (23)
Indian Statistical Institute, Kolkata, India

Neshat Mohammadi (45)
Department of Computer Science, University of Illinois at Chicago, IL, USA

Jonathan Mosheiff (9)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Lukas Nölke (37)
Department for Mathematics and Computer Science, University of Bremen, Germany

Maciej Obremski (1)
National University of Singapore, Singapore

Jakub Opršal (34)
Computer Science Department, Durham University, UK

Anurag Pandey (8)
Max Planck Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany

Fahad Panolan (32)
Department of Computer Science and Engineering, IIT Hyderabad, India

Manaswi Paraashar (23)
Indian Statistical Institute, Kolkata, India

Ojas Parekh (59)
Sandia National Laboratories, Albuquerque, NM, USA

Pan Peng (16)
Department of Computer Science, University of Sheffield, UK

Jeff M. Phillips (12)
School of Computing, University of Utah, Salt Lake City, UT, USA

Madhusudhan Reddy Pittu (47)
Indian Institute of Technology, Kharagpur, India

Adam Polak (52)
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland

Ely Porat (46)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Aaron Potechin (58)
University of Chicago, IL, USA

Aditya Potukuchi (30)
Department of Computer Science, Rutgers University, Piscataway, NJ, USA

Ali Pourmiri (11)
Macquarie University, Sydney, Australia

Eric Price (13)
Department of Computer Science, University of Texas at Austin, TX, USA

Yuval Rabani (54)
Hebrew University of Jerusalem, Israel

Cyrus Rashtchian (26)
Department of Computer Science & Engineering, UC San Diego, CA, USA

Malin Rau (44)
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP*, LIG, Grenoble, France

Ran Raz (21)
Department of Computer Science, Princeton University, NJ, USA

Oded Regev (43)
Courant Institute of Mathematical Sciences, New York University, NY, USA

Nicolas Resch (9)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

João Ribeiro (1)
Imperial College London, UK

Dana Ron (27)
Tel Aviv University, Israel

Asaf Rosin (27)
Tel Aviv University, Israel

Peter Rossmanith (14)
Department of Computer Science, RWTH Aachen University, Germany

Sai Sandeep (34, 40)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Thatchaphol Saranurak (33)
Toyota Technological Institute at Chicago, IL, USA

Kanthi Sarpatwar (48)
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA

Saket Saurabh (51)
The Institute of Mathematical Sciences, HBNI, Chennai, India; University of Bergen, Norway

Jonathan Scarlett (13)
Department of Computer Science & Department of Mathematics, National University of Singapore, Singapore

Baruch Schieber (48)
Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA

Roy Schwartz (49, 53)
Department of Computer Science, Technion, Haifa, Israel

Francesco Sgherzi (45)
Department of Computer Science, University of Illinois at Chicago, IL, USA

Hadas Shachnai (48)
Computer Science Department, Technion, Haifa, Israel

Ronen Shaltiel (10)
University of Haifa, Israel

Ella Sharakanski (25)
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Yotam Sharoni (53)
Department of Computer Science, Technion, Haifa, Israel

Anastasios Sidiropoulos (36, 45)
Department of Computer Science, University of Illinois at Chicago, IL, USA

Shashwat Silas (9)
Department of Computer Science, Stanford University, CA, USA

Kirill Simonov (32)
Department of Informatics, University of Bergen, Norway

Christian Sohler (16)
Department of Mathematics and Computer Science, University of Cologne, Germany

Tatiana Starikovskaya (35)
DIENS, École normale supérieure, PSL Research University, Paris, France

Noah Stephens-Davidowitz (1)
Cornell University, Ithaca, NY, USA

Amanda Pascoe Streib (3)
Center for Computing Sciences, Bowie, MD, USA

Noah Streib (3)
Center for Computing Sciences, Bowie, MD, USA

Michal Svagerka (35)
ETH Zürich, Switzerland

Zoltán Szigeti (61)
University Grenoble Alpes, CNRS, G-SCOP, France

Amnon Ta-Shma (7)
The Blavatnik School of Computer Science, Tel-Aviv University, Israel

Wai Ming Tai (12)
School of Computing, University of Utah, Salt Lake City, UT, USA

Prafullkumar Tale (51)
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany

Yi Tang (43)
Courant Institute of Mathematical Sciences, New York University, NY, USA

Roei Tell (7)
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel

Justin Thaler (22)
Georgetown University, Washington, DC, USA

Theophile Thiery (62)
School of Mathematical Sciences, Queen Mary University of London, UK

Linh Tran (20)
Department of Mathematics, Yale University, New Haven, CT, USA

Przemysław Uznański (35, 46)
Institute of Computer Science, University of Wrocław, Poland

Mina Valizadeh (45)
Department of Computer Science, University of Illinois at Chicago, IL, USA

Daniel Vaz (39)
Operations Research Group, TU Munich, Germany

Van Vu (20)
Department of Mathematics, Yale University, New Haven, CT, USA

Thuy-Duong Vuong (56)
Department of Computer Science, Stanford University, CA, USA

Lu Wang (61)
Department of Combinatorics and Optimization, University of Waterloo, Canada

Justin Ward (62)
School of Mathematical Sciences, Queen Mary University of London, UK

Osamu Watanabe (15)
Tokyo Institute of Technology, Japan

Thomas Watson (28)
University of Memphis, TN, USA

Alexander Wei (60)
Harvard University, Cambridge, MA, USA

Hao-Ting Wei (63)
Department of IEOR, Columbia University, New York, NY, USA

Karl Wimmer (24)
Duquesne University, Pittsburgh, PA, USA

David P. Woodruff (26, 50, 64)
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA

Mary Wootters (9)
Department of Computer Science, Stanford University, CA, USA

Jiayi Xian (39, 42)
Department of Computer Science and Engineering, University at Buffalo, NY, USA

Chao Xu (17)
The Voleon Group, Berkeley, CA, USA

Yuhao Zhang (63)
Department of Computer Science, The University of Hong Kong, China

Hanlin Zhu (26)
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China

Extractor Lower Bounds, Revisited

Divesh Aggarwal
National University of Singapore, Singapore
dcsdiva@nus.edu.sg

Siyao Guo
New York University Shanghai, China
siyao.guo@nyu.edu

Maciej Obremski
National University of Singapore, Singapore
obremski.math@gmail.com

João Ribeiro
Imperial College London, UK
j.lourenco-ribeiro17@imperial.ac.uk

Noah Stephens-Davidowitz
Cornell University, Ithaca, NY, USA
noahsd@gmail.com

Abstract

We revisit the fundamental problem of determining seed length lower bounds for strong extractors and natural variants thereof. These variants stem from a "change in quantifiers" over the seeds of the extractor: While a strong extractor requires that the average output bias (over all seeds) is small for all input sources with sufficient min-entropy, a *somewhere* extractor only requires that there *exists* a seed whose output bias is small. More generally, we study what we call *probable* extractors, which on input a source with sufficient min-entropy guarantee that a large enough fraction of seeds have small enough associated output bias. Such extractors have played a key role in many constructions of pseudorandom objects, though they are often defined implicitly and have not been studied extensively.

Prior known techniques fail to yield good seed length lower bounds when applied to the variants above. Our novel approach yields significantly improved lower bounds for somewhere and probable extractors. To complement this, we construct a somewhere extractor that implies our lower bound for such functions is tight in the high min-entropy regime. Surprisingly, this means that a random function is far from an optimal somewhere extractor in this regime. The techniques that we develop also yield an alternative, simpler proof of the celebrated optimal lower bound for strong extractors originally due to Radhakrishnan and Ta-Shma (SIAM J. Discrete Math., 2000).

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization; Theory of computation → Expander graphs and randomness extractors

Keywords and phrases randomness extractors, lower bounds, explicit constructions

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.1

Category RANDOM

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/173/` [1].

Funding *Divesh Aggarwal*: This work is supported in part by the Singapore National Research Foundation under NRF RF Award No. NRF-NRFF2013-13, the Ministry of Education, Singapore under the Research Centres of Excellence programme by the Tier-3 grant Grant "Random numbers from quantum processes" No. MOE2012-T3-1-009.
Siyao Guo: Supported by Shanghai Eastern Young Scholar Program. Part of the work was done while visiting the Centre for Quantum Technologies, National University of Singapore.



© Divesh Aggarwal, Siyao Guo, Maciej Obremski, João Ribeiro, and Noah Stephens-Davidowitz;
licensed under Creative Commons License CC-BY



Maciej Obremski: This author was supported by MOE2019-T2-1-145 Foundations of quantum-safe cryptography.
João Ribeiro: Part of the work was done while visiting the Centre for Quantum Technologies, National University of Singapore.
Noah Stephens-Davidowitz: Parts of this work were done while visiting the Simons Institute for the Theory of Computing, Massachusetts Institute of Technology, and the Centre for Quantum Technologies, National University of Singapore.

Acknowledgements The authors wish to thank the anonymous reviewers for several comments that improved the paper.

1 Introduction

Strong seeded extractors are central objects in pseudorandomness that have found many applications in theoretical computer science and cryptography. Informally speaking, a function $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ is a strong extractor if for every source X of sufficiently high min-entropy it holds that the average bias of $\mathsf{Ext}(X,i)$ over the seeds $i \in [D]$ is small. More precisely, we have the definition below. Throughout this paper, we focus on single-bit output extractors since lower bounds in this setting immediately imply lower bounds for any m-bit output extractor.

▶ **Definition 1.1** ((k,ε)-strong extractor). *For $\varepsilon < 1/2$, a function $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ is said to be a (k,ε)-*strong extractor *if*

$$\mathop{\mathbb{E}}_{i \leftarrow [D]} [\Delta(\mathsf{Ext}(X,i); U_1)] \leq \varepsilon \tag{1}$$

for every (n,k)-source[1] X, where $i \leftarrow [D]$ means i is uniformly distributed over $[D]$, and

$$\Delta(\mathsf{Ext}(X,i); U_1) = |\Pr[\mathsf{Ext}(X,i) = 1] - 1/2|$$

is the bias of $\mathsf{Ext}(X,i)$.

A fundamental parameter when studying strong extractors is the number of seeds D. Ideally, one would like to construct strong extractors with D as small as possible. However, there exist lower bounds on D depending on n, k, and ε. Nisan and Zuckerman [21] showed that every strong extractor must use $D = \Omega\left(\frac{n-k}{\varepsilon}\right)$ seeds. Later, in a seminal work, Radhakrishnan and Ta-Shma [22] improved the lower bound above to[2]

$$D = \Omega\left(\frac{n-k}{\varepsilon^2}\right). \tag{2}$$

Notably, this turns out to be tight. In fact, a random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ with $D = C \cdot \frac{n-k}{\varepsilon^2}$ seeds, for a sufficiently large constant $C > 0$, is a (k,ε)-strong extractor with high probability. An alternative proof of (2) for a limited range of k was given by Bar-Yossef, Kumar, and Sivakumar [3], based on the connection between extractors and averaging samplers [27] and sampling lower bounds.

[1] See Definition 2.3.
[2] The lower bound in (2) also holds for "non-strong" extractors, i.e., functions $F : \{0,1\}^n \times [D] \to \{0,1\}^{d+1}$ such that $F(X, U_{[D]}) \approx_\varepsilon U_{d+1}$, where $d = \log D$. Note that in this case the lower bound only holds for output length at least $d+1$; otherwise, one can just output the uniformly random seed.

At the opposite end of the spectrum lies another well-known pseudorandom object, called a *somewhere extractor*. While a strong extractor has small average bias, all we require of a somewhere extractor is that its *minimum* bias over all seeds is small. More precisely, we have the following definition.

▶ **Definition 1.2** ((k,ε)-somewhere extractor)**.** *For $\varepsilon < 1/2$, a function* $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ *is said to be a* (k,ε)-somewhere extractor *if for every (n,k)-source X it holds that*

$$\min_{i\in[D]} \Delta(\mathsf{Ext}(X,i);U_1) \leq \varepsilon.$$

Somewhere extractors arise in a number of different contexts. In fact, many of the most important applications of strong extractors (e.g., in the construction of multi-source extractors) actually only require these potentially weaker objects. Given the complete picture we have of strong extractors, it is natural to wonder what kind of bounds we can prove on the number of seeds D for a somewhere extractor Ext.

A simple averaging argument on the preimage sizes of Ext shows that $D > n - k$, but it is possible to improve on this lower bound. If one considers somewhere extractors with m output bits, then [2] showed that a connection to dispersers leads to the lower bound

$$D = \Omega\left(\frac{n-k}{\varepsilon + 2^{-m}}\right). \tag{3}$$

While (3) was good enough in the context of [2], it is quite unsatisfactory in general for two reasons: First, it is trivial for small m (e.g., in our setting, where $m = 1$). Second, even for larger m, it does not scale with ε below 2^{-m}. In the 1-bit output setting, which is the hardest for lower bounds, the best known lower bound is given in [2] as

$$D = \Omega\left(n - k + \log\left(1/\varepsilon\right)\right).$$

On the other hand, as we show in this work, a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ is a (k,ε)-somewhere extractor with non-negligible probability only when $D = \Omega\left(\frac{n-k}{\varepsilon^2}\right)$ (we discuss this in more detail below). Therefore, in contrast with strong extractors, there is a large gap between upper and lower bounds on the number of seeds required by somewhere extractors, leaving open the exciting possibility of better constructions.

One may also wonder whether the strong extractor lower bound techniques from [21, 22] can be adapted to yield better lower bounds for somewhere extractors. However, it is not clear how this can be done, since these techniques are fundamentally tailored for dealing solely with the average bias as in Definition 1.1. Overall, current techniques seem incapable of yielding a sharp, unconditional analysis of somewhere extractors.

1.1 Our contributions

In this work, we develop a novel approach towards proving lower bounds on the number of seeds required by natural variants of strong extractors. We highlight our main results here.

1.1.1 Improved lower bounds for somewhere extractors

We significantly improve the lower bound for (k,ε)-somewhere extractors. More precisely, we prove the following result.

▶ **Theorem 1.3.** *Every (k,ε)-somewhere extractor* $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ *must have*

$$D \geq \frac{\ln 2}{2} \cdot \frac{n-k}{\varepsilon}. \tag{4}$$

Recall that the previous best lower bound was $D = \Omega(n - k + \log(1/\varepsilon))$. Observe also that the lower bound in (4) is a factor of ε smaller than the one in (2) for strong extractors. Remarkably, we construct a (simple) (k, ε)-somewhere extractor that shows (4) is *tight* in the high min-entropy regime.

▶ **Theorem 1.4.** *For every $\varepsilon \geq \frac{1}{2(1+2^k)}$, there exists a (k, ε)-somewhere extractor* $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ *with*

$$D = \frac{2^{n-k-1}}{\varepsilon} + 1.$$

▶ Remark 1.5. We note that there are no (k, ε)-somewhere extractors with $\varepsilon < \frac{1}{2(1+2^k)}$. To see this, consider an (n, k)-source X uniformly distributed over a set of size $2^k + 1$. Then, for every i, $\Pr[\mathsf{Ext}(X, i) = 1] = \frac{C}{2^k+1}$ for some integer $C \geq 0$, and so $\mathsf{Ext}(X, i)$ has bias $\left|\frac{C}{2^k+1} - \frac{1}{2}\right| = \left|\frac{C - 2^{k-1} - 1/2}{2^k+1}\right| \geq \frac{1}{2(1+2^k)}$

In particular, Theorem 1.4 shows that (4) is tight up to a constant factor when $n - k$ is constant.

On the other hand, the existential result for strong extractors immediately implies that a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ with $D = C \cdot \frac{n-k}{\varepsilon^2}$ for a large enough constant $C > 0$ is a (k, ε)-somewhere extractor with high probability. Interestingly, we show that this probabilistic argument is tight up to a constant factor, in the sense that a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ must have $D = \Omega\left(\frac{n-k}{\varepsilon^2}\right)$ in order to be a (k, ε)-somewhere extractor with non-negligible probability for essentially all regimes of k and ε. Given the above, we conclude that a random function is far from an optimal (k, ε)-somewhere extractor in the high min-entropy regime. This provides a rare example where an explicit construction actually yields a significantly better extractor than a random function (at least for some parameters), and highlights a qualitative difference with strong extractors.

To be more precise, we show the following.

▶ **Theorem 1.6.** *For large enough n, suppose that $k \leq n - 400$, $2^{-0.24(n+k)} \leq \varepsilon \leq c$ for a sufficiently small constant $c > 0$, and*

$$D \leq \frac{n-k}{400 \cdot \varepsilon^2}.$$

Then, a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ is not *a (k, ε)-somewhere extractor with probability at least $1 - 2^{-2^{\Omega(n)}}$.*

1.1.2 Simple proof of the optimal lower bound for strong extractors

In the setting of strong extractors, we give an alternative proof of the tight lower bound (2). Our proof is much simpler than those due to Radhakrishnan and Ta-Shma [22] and Bar-Yossef, Kumar, and Sivakumar [3]. To be precise, we prove the following result.

▶ **Theorem 1.7.** *For every $n, k, \varepsilon > 0$ satisfying $n - k \geq 39$ and every (k, ε)-strong extractor* $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ *it holds that*

$$D \geq \frac{\ln 2}{18} \cdot \frac{n-k}{\varepsilon^2}.$$

1.1.3 Generalizing somewhere extractors and lower bounds

Finally, we initiate the systematic study of a meaningful generalization of somewhere extractors and also obtain significantly improved lower bounds in that setting, as discussed below. A somewhere extractor Ext can be generalized in a natural way by requiring that some fraction of the seeds of Ext yield an unbiased output, instead of only a single seed. This leads to the following definition.

▶ **Definition 1.8** ((k, ε, δ)-probable extractor)**.** *For $\varepsilon < 1/2$, a function $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ is said to be a (k, ε, δ)-*probable extractor *if*

$$\Pr_{i \leftarrow [D]}[\Delta(\mathsf{Ext}(X, i); U_1) > \varepsilon] < \delta$$

for every (n, k)-source X.

We note that probable extractors have been defined explicitly before, but not studied in depth, in [23, 6]. Observe that a (k, ε)-somewhere extractor corresponds to a $(k, \varepsilon, \delta = 1)$-probable extractor. Moreover, a (k, ε)-strong extractor lies somewhere between a $(k, \varepsilon/2, \varepsilon/2)$-probable extractor and a $(k, \sqrt{\varepsilon}, \sqrt{\varepsilon})$-probable extractor. More generally, every (k, ε)-strong extractor is a $(k, \varepsilon/\delta, \delta)$-probable extractor for every $\delta > 0$ by Markov's inequality. On the other hand, we also have that every (k, ε, δ)-probable extractor is a $(k, \varepsilon + \delta)$-strong extractor.

Given our previous discussion, a natural question to ask about probable extractors is the following:

How do ε and δ influence the number of seeds D?

Our work leads to a better understanding of this behavior. Similarly to what was already discussed in [6], by separating the maximum fraction of "bad" seeds δ and the maximum bias of the "good" seeds ε, we are able to explore the explicit influence that each of these parameters has on the number of seeds. Such a fine-grained analysis is not possible, for example, in the case of strong extractors, since those properties are essentially merged into a single global error parameter.

Besides being interesting on its own, there are practical motivations for the question above. In fact, several constructions of multi-source extractors make use of (k, ε)-strong extractors in scenarios where a $(k, \varepsilon/\delta, \delta)$-probable extractor would suffice with δ much larger than ε. The reason for this is simply that no better constructions of $(k, \varepsilon/\delta, \delta)$-probable extractors are known. However, it could be a priori possible to design a $(k, \varepsilon/\delta, \delta)$-probable extractor requiring much fewer seeds than a (k, ε)-strong extractor. In turn, this would lead to simpler constructions of, and improved parameters for, several multi-source extractors. We expand on this in Section 1.2.

1.1.4 Lower bounds for probable extractors

Lower bounds on the number of seeds required by probable extractors can be derived directly from lower bounds for both strong and somewhere extractors. Combining (2) with the fact that every (k, ε, δ)-probable extractor is a $(k, \varepsilon + \delta)$-strong extractor immediately leads to the lower bound

$$D = \Omega\left(\frac{n-k}{(\varepsilon+\delta)^2}\right).$$

However, note that the bound above becomes trivial whenever one of ε or δ is large.

To achieve a stronger bound, we observe that a (k, ε, δ)-probable extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ must be a (k, ε)-somewhere extractor when restricted to the first δD seeds. Therefore, any lower bound L for the number of seeds of (k, ε)-somewhere extractors immediately implies the lower bound $D \geq L/\delta$ for any (k, ε, δ)-probable extractor. Combining this with Theorem 1.3 leads to the following result.

▶ **Theorem 1.9.** *Let* $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ *be a* (k, ε, δ)*-probable extractor. Then, it holds that*

$$D \geq \frac{\ln 2}{2} \cdot \frac{n-k}{\varepsilon \cdot \delta}. \tag{5}$$

The lower bound in (5) significantly improves upon all previous bounds over a large range of (ε, δ), namely when $\delta \gg \varepsilon$ or $\varepsilon \gg \delta$. On the other hand, we show that a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ with $D = O\left(\frac{n-k}{\varepsilon^2 \cdot \delta}\right)$ is a (k, ε, δ)-probable extractor with high probability. It remains an open problem to close the gap between this upper bound and Theorem 1.9 in general. While we know from our previous discussion that the lower bound in Theorem 1.9 is tight for $\delta = 1$ and $n - k = O(1)$, the behavior might change substantially for other parameters.

Given the gap between the bounds above, it is natural to ask whether a different probabilistic argument could be used to show that a uniformly random function using fewer seeds is a (k, ε, δ)-probable extractor with high probability. As before, we can easily extend Theorem 1.6 to the setting of probable extractors to show the answer to the question above is negative. Namely, we have the following result, which shows that our probabilistic construction is tight up to a constant factor.

▶ **Theorem 1.10.** *For any* $\delta = \delta(n) \in (0, 1]$ *and large enough* n*, suppose that* $k \leq n - 400$*,* $2^{-0.24(n+k)} \leq \varepsilon \leq c$ *for a sufficiently small constant* $c > 0$*, and*

$$D \leq \frac{n-k}{400 \cdot \varepsilon^2 \cdot \delta}.$$

Then, a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ *is* not *a* (k, ε, δ)*-probable extractor with probability at least* $1 - 2^{-2^{\Omega(n)}}$*.*

1.2 Applications of somewhere- and probable- extractors

Besides the works we have already discussed, several others have either implicitly or explicitly used probable extractors. Many constructions of seeded and multi-source extractors [26, 20, 25, 24, 15, 17, 16, 9, 18, 19, 6, 8], along with some constructions of dispersers [5] and non-malleable and affine extractors [7, 19], use probable extractors (or slight variants of probable extractors) in their constructions.

In the literature, the output of the probable extractor (concatenated over all D seeds) is usually called a *somewhere-random source with D rows*. The time complexity of the resulting extractor constructions depends linearly on the complexity of enumerating the D seeds of the probable extractor being used. This poses a problem because, even now, the best explicit probable extractor we know of for a single weak source is simply a strong extractor. For such constructions, the lower bound in (2) applies, and so extra assumptions must be made or parameters must be worsened in order to ensure that seed enumeration can be done efficiently.

We present concrete examples of the compromise above. Some works settle for a large overall $1/\mathrm{poly}(n)$ error of the resulting extractor to get around the seed enumeration problem [24, 15, 17, 16, 19]. On another front, many works use extra independent weak sources with enough min-entropy as input to generate somewhere-random sources with fewer rows [4, 5, 9, 18, 19, 7, 6, 8]. Moreover, the addition of a short uniformly random seed to

achieve this goal has also been considered [20]. Many works above can be interpreted as constructing several types of randomness extractors for somewhere-random sources (called *mergers*), a problem which was first studied by Ta-Shma [26]. Other works that have studied mergers include [25, 28, 12, 11, 13, 10].

Prior to this work, we could not rule out a (k, ε, δ)-probable extractor for δ much larger than ε with much fewer seeds than a (k, ε)-strong extractor. Given the discussion above, this would lead not only to extractors with improved parameters, but also to conceptually simpler constructions, since many tools and assumptions were introduced to deal with the fact that somewhere-random sources generated by strong extractors have rather many rows. Our results preclude this possibility.

Finally, we note that many of the applications above still work if one considers an extractor that outputs *convex combinations* of somewhere-random sources from (n, k)-sources instead. Our lower bounds do not apply to this weaker setting. Therefore, we do not rule out the existence of methods of generating a convex combination of somewhere-random sources from one weak source requiring fewer seeds. Our results show that, without considering convex combinations, one cannot do *too much* better than the naive and globally used method of enumerating over the seeds of a strong extractor (although, surprisingly, we show that a polynomial improvement in $1/\varepsilon$ *is* possible). We leave it as an interesting open problem to extend our new techniques and bounds to the setting of convex combinations.

1.3 Technical overview

In this section, we provide a more detailed account of our contributions.

1.3.1 The high-level approach

Our extractor lower bounds can be unified under a common high-level approach. Fix an arbitrary function $F : \{0,1\}^n \times [D] \to \{0,1\}$. Our goal is to relate the number of seeds D to some measure of the bias of F over all seeds, depending on the type of extractor we are dealing with. For the remainder of this section, we focus on somewhere extractors and the minimum bias. However, everything is equally applicable to strong extractors simply by replacing minimum bias by average bias.

To prove a lower bound on D, we must show the existence of an input (n, k)-source X such that $P = (F(X, 1), F(X, 2), \ldots, F(X, D))$ is sufficiently biased. We do this by constructing the output distribution P directly, rather than trying to find an input distribution X that maps to P.

Throughout the remainder of this paper, we write

$$F(x) = (F(x, 1), F(x, 2), \ldots, F(x, D)) \in \{0,1\}^D.$$

Then, for F and fixed k, we construct a distribution $P = (P_1, P_2, \ldots, P_D)$ over $\{0,1\}^D$ such that the following two conditions hold.

1. There exists an (n, k)-source X such that $F(X) = P$;
2. It holds that[3]

$$\min_{i \in [D]} \Delta(P_i; U_1) \geq \alpha,$$

where α is some quantity depending on n, k, and D.

If F is a (k, ε)-somewhere extractor, the two conditions above imply that $\varepsilon \geq \alpha$. This relationship then yields a lower bound on D.

[3] In the case of strong extractors, this condition is replaced by $\mathbb{E}_{i \leftarrow [D]}[\Delta(P_i; U_1)] \geq \alpha$.

The main novelty of our approach lies in the design of the output distribution P. The distribution $A = F(U_n)$ takes on a special role in our construction of good choices of P. We begin by showing that the first condition above automatically holds provided P satisfies a simple constraint related to A, which is detailed in the following lemma (below and throughout the paper, we write $X(x)$ for the probability that a random variable/distribution X takes on value x).

▶ **Lemma 1.11.** *There exists an (n,k)-source X such that $F(X) = P$ if*

$$P(a) \le 2^{n-k} A(a) \tag{6}$$

for all $a \in \{0,1\}^D$.

Proof. It is enough to consider the source $X \in \{0,1\}^n$ that picks each $x \in \{0,1\}^n$ with probability

$$X(x) = 2^{-n} \cdot \frac{P(F(x))}{A(F(x))}.$$

First, by (6) it follows that $X(x) \le 2^{-n} \cdot 2^{n-k} = 2^{-k}$ for every x. Moreover, using the fact that $A(a) = 2^{-n} \cdot |F^{-1}(a)|$, it is easy to see that X is a valid probability distribution and $F(X) = P$. ◀

▶ Remark 1.12. *It is easy to see that such an (n,k)-source exists if* and only if (6) *holds. However, in this work, we only need the implication in one direction.*

We construct distributions P implicitly in terms of the distribution $A = F(U_n)$. In fact, Lemma 1.11 shows it is enough to restrict our attention to distributions P that can be written as

$$P(a) = A(a) \cdot f(a)$$

for some non-negative function f satisfying $f(a) \le 2^{n-k}$ for all $a \in \{0,1\}^D$. As discussed in the following sections, careful choices of f lead to good lower bounds on D with streamlined derivations.

1.3.2 Improved lower bound for somewhere and probable extractors

We employ the high-level approach detailed in Section 1.3.1 to obtain improved lower bounds for probable extractors. Namely, we prove Theorem 1.3, which states that every (k,ε)-somewhere extractor must have $D \ge \frac{\ln 2}{2} \cdot \frac{n-k}{\varepsilon}$. In turn, this easily implies Theorem 1.9 for general (k,ε,δ)-probable extractors.

Let $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ be an arbitrary (k,ε)-somewhere extractor. In order to prove Theorem 1.3 via the high-level approach from Section 1.3.1, we consider the family of distributions $\overline{P}_z$ parameterized by $z \in \{0,1\}^D$ defined as

$$\overline{P}_z(a) = \frac{1}{C_z} \cdot A(a) \cdot \prod_{i=1}^{D} [1 + (-1)^{a_i + z_i} \gamma], \quad a \in \{0,1\}^D,$$

where $A = \mathsf{Ext}(U_n)$, $\gamma \in (0,1)$ is a parameter of our choice, and C_z is the normalizing factor.

We choose $z^\star$ which maximizes C_z over all $z \in \{0,1\}^D$, and consider $P = \overline{P}_{z^\star}$. In particular, this choice implies that $C_{z^\star} \ge 1$, which allows us to take $\gamma = \Theta\left(\frac{n-k}{D}\right)$ while still satisfying (6). It remains to lower bound $\Delta(P_i; U_1)$ for every $i \in [D]$ appropriately. The product structure of the family of distributions we consider makes it amenable to a Fourier-analytic approach, which we employ to show that for every $i \in [D]$ we have

$$\Delta(P_i; U_1) = \Omega(\gamma) = \Omega\left(\frac{n-k}{D}\right).$$

This yields the desired lower bound on D. More details can be found in Section 3.

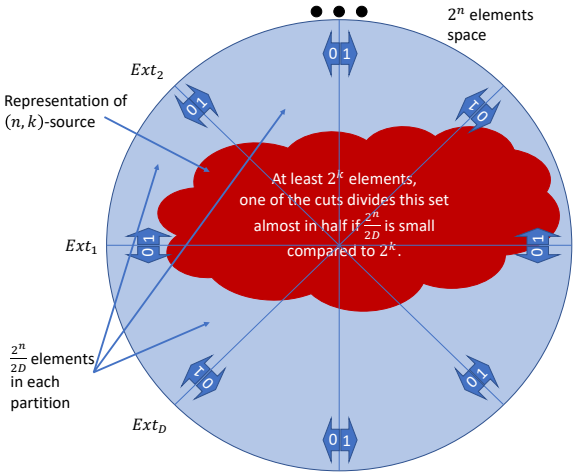


Figure 1 An illustration of the (k, ε)-somewhere extractor that shows our lower bound is tight in the high min-entropy regime.

1.3.3 Tight upper bound for somewhere extractors

We design a somewhere extractor that shows our lower bound for (k, ε)-somewhere extractors is tight (up to a multiplicative constant) in the high min-entropy regime where $n - k = O(1)$. More precisely, we prove Theorem 1.4, which states that there exists a (k, ε)-somewhere extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ with $D = \frac{2^{n-k-1}}{\varepsilon} + 1$ for all non-trivial ε.

This is accomplished by showing that the function $\mathsf{Ext} : [N] \times \{0, 1, \ldots, E\} \to \{0,1\}$, with $E = \frac{2^{n-k-1}}{\varepsilon}$ and $N = 2^n$, defined as

$$\mathsf{Ext}(x, i) = \mathsf{sign}[(x + i) \mod 2E], \quad i = 0, 1, \ldots, E \tag{7}$$

is a (k, ε)-somewhere extractor. In (7), we see $x \mod 2E$ as an integer in $\{-E, \ldots, E-1\}$, and define $\mathsf{sign}(y) = \mathbf{1}_{\{y \geq 0\}}$. Intuitively, this simple function yields a good somewhere extractor because the functions $\mathsf{Ext}(\cdot, i)$ "transition smoothly" from $\mathsf{Ext}(\cdot, 0)$ to its opposite, $\mathsf{Ext}(\cdot, E) = 1 - \mathsf{Ext}(\cdot, 0)$, as shown in Figure 1.

In more detail, given an (n, k)-source X, we wish to prove that there is a seed i such that $\Delta(\mathsf{Ext}(X, i); U_1) \leq \varepsilon$. In order to show this, we will look at how the quantities

$$\Delta_i = \Pr[\mathsf{Ext}(X, i) = 1] - \Pr[\mathsf{Ext}(X, i) = 0], \quad i = 0, 1, \ldots, E$$

behave. The desired result follows if we show that $|\Delta_i| \leq 2\varepsilon$ for some i. In turn, this holds because the Δ_i's satisfy two simple properties. First, we have $\Delta_0 = -\Delta_E$. Second, when going from $\mathsf{Ext}(\cdot, i-1)$ to $\mathsf{Ext}(\cdot, i)$, by our choice of parameters at most $4\varepsilon \cdot 2^k$ elements of $[N]$ go from 1 to 0, and vice-versa. This implies that $|\Delta_i - \Delta_{i-1}| \leq 4\varepsilon$. Combining the two properties above immediately ensures the existence of $i^\star$ such that $|\Delta_{i^\star}| \leq 2\varepsilon$, as desired. For more details, see Section 4.

1.3.4 Simpler proof of the optimal lower bound for strong extractors

In this section, we discuss our alternative, simpler proof of the optimal lower bound on the number of seeds for strong extractors, originally obtained by Radhakrishnan and Ta-Shma [22]. Namely, we prove Theorem 1.7, which states that every (k, ε)-strong extractor must have $D \geq \frac{\ln 2}{18} \cdot \frac{n-k}{\varepsilon^2}$ when $n - k \geq 39$.

As before, we follow the high-level approach introduced in Section 1.3.1. However, we consider a different family of distributions. Fix an arbitrary (k, ε)-strong extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$, and let $A = \mathsf{Ext}(U_n)$. Then, for $z \in \{0,1\}^D$ and $t \leq D$ define the distribution

$$\overline{P}_{z,t}(a) = \frac{1}{C_{z,t}} \cdot A(a) \cdot \mathbf{1}_{\{\|z-a\|_1 \leq t\}}, \quad a \in \{0,1\}^D,$$

where $C_{z,t}$ is the normalizing factor. The desired result now follows via two simple combinatorial arguments, which guarantee that (i) for an appropriate $t = D/2 - \Theta(\sqrt{(n-k)D})$, there exists a choice of z such that $P = \overline{P}_{z,t}$ satisfies (6), and (ii) the average bias of every distribution $\overline{P}_{z,t}$ is at least $1/2 - t/D = \Omega\left(\sqrt{(n-k)/D}\right)$. More details can be found in Appendix B.

1.3.5 Probabilistic constructions and lower bounds for random functions

We study for which values of D it holds that a uniformly random function $F : \{0,1\}^n \times [D] \to \{0,1\}$ is a (k, ε, δ)-probable extractor with non-negligible probability. To show an upper bound, we consider a connection between probable extractors and strong *two-source* extractors, and then invoke well-known existential results for the latter.[4] This shows that, under a mild constraint on k, ε, and δ, a uniformly random function is a (k, ε, δ)-probable extractor with probability at least, say, 0.99 when $D = C \cdot \frac{n-k}{\varepsilon^2 \cdot \delta}$ for a sufficiently large constant $C > 0$.

We complement the upper bound in the previous paragraph via Theorem 1.10, which states that a uniformly random function with $D \leq \frac{n-k}{400 \cdot \varepsilon^2 \cdot \delta}$ is *not* a (k, ε, δ)-probable extractor with probability at least $1 - 2^{-2^{\Omega(n)}}$. This means that our probabilistic construction above is tight up to a constant factor. Similarly to Section 1.3.2, to prove this result it suffices to focus our attention on somewhere extractors. We consider a source $X \in \{0,1\}^n$ uniformly distributed over a set $\mathcal{X}_F$ defined as

$$\mathcal{X}_F = \{x \in \{0,1\}^n : \|F(x)\|_1 \leq t\}$$

for an appropriate $t = D/2 - \Theta(\sqrt{(n-k)D})$. Then, we show X satisfies two properties: First, by a Chernoff bound, it holds that $|\mathcal{X}_F| \geq 2^k$ with very high probability over the choice of F, and hence X is an (n, k)-source with very high probability. Second, we show that, again with very high probability, we have $\Delta(F(X, i); U_1) > \varepsilon$ simultaneously for all $i \in [D]$. These two properties immediately imply that F is not a (k, ε)-somewhere extractor with high probability. More details can be found in the full version of this paper [1].

1.4 Open questions

Besides the natural problem of improving upon our lower bounds in general, our work leaves open other interesting avenues for further research.:

- Consider the special case of $(k, \varepsilon, \varepsilon)$-probable extractors. In this setting, the best lower bounds only yield $D = \Omega\left(\frac{n-k}{\varepsilon^2}\right)$, while the probabilistic method requires $D = \Omega\left(\frac{n-k}{\varepsilon^3}\right)$ to work with non-negligible probability. We believe $D = \Theta\left(\frac{n-k}{\varepsilon^3}\right)$ is the correct answer, and it would be very interesting to prove (or disprove) this claim, as it showcases different behavior than (k, ε, δ)-probable extractors for $\delta \gg \varepsilon$;

[4] More direct approaches do not seem to work because the set of sources from which a somewhere extractor successfully extracts is not necessarily convex.

- Show the existence of a (k, ε)-somewhere extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ using $D = O\left(\frac{n-k}{\varepsilon}\right)$ seeds. This would extend the tightness of our lower bound for somewhere extractors below the high min-entropy regime;
- Extend our (k, ε)-somewhere extractor from Section 1.3.3 to output $m > 1$ bits with (roughly) the same number of seeds;
- Extend our lower bounds to the setting where one is allowed to output convex combinations of somewhere-random sources from one (n, k)-source (see Section 1.2).

1.5 Organization

We introduce basic notions and results that are useful throughout our work in Section 2. The proofs of our extractor lower bounds are presented in Section 3 and Appendices A and B. The matching upper bound on the number of seeds of somewhere extractors can be found in Section 4. Probabilistic constructions of probable extractors, along with lower bounds on the number of seeds of uniformly random functions are discussed in the full version [1].

2 Preliminaries

2.1 Notation

Random variables and distributions are usually denoted by uppercase letters such as X, Y, and Z. When context allows, we may confuse a random variable with its associated distribution. We write $X(x)$ for the probability that X equals x, and denote the support of X by $\mathsf{supp}(X)$. The uniform distribution over $\{0,1\}^n$ is denoted by U_n. We write $i \leftarrow S$ to mean that i is sampled uniformly at random from the set S. For a distribution X, we write $x \sim X$ to denote x is sampled according to X. Given an event E, the indicator of E is denoted by $\mathbf{1}_{\{E\}}$. The expected value of a random variable X is denoted by $\mathbb{E}[X]$ or $\mathbb{E}_{x\sim X}[x]$. Sets are usually denoted by uppercase letters such as S and T. The set $\{1, 2, \ldots, D\}$ is denoted by $[D]$. We will usually identify a set S with its characteristic vector, so that we write $S_i = 1$ if and only if $i \in S$. We write $S + T$ for the symmetric difference between two sets S and T (i.e., the modulo 2 sum of their characteristic vectors). We denote the base-2 logarithm by log and the natural logarithm by ln. We write $\|x\|_p$ for the p-norm of a vector x. The inner product between two vectors x and y over some field is denoted by $\langle x, y\rangle$.

2.2 Probability theory

In this section, we introduce some basic notions and results from probability theory.

▶ **Definition 2.1** (Statistical distance). *Given two distributions X and Y over a set $\mathcal{X}$, the* statistical distance between X and Y, *denoted by $\Delta(X;Y)$, is defined as*

$$\Delta(X;Y) = \max_{S\subseteq\mathcal{X}} |\Pr[X \in S] - \Pr[Y \in S]| = \frac{1}{2}\sum_{x\in\mathcal{X}} |X(x) - Y(x)| .$$

*We say that X and Y are ε-*close*, also written $X \approx_\varepsilon Y$, if $\Delta(X;Y) \le \varepsilon$.*

▶ **Definition 2.2** (Min-entropy). *Given a distribution X over $\mathcal{X}$, the* min-entropy of X, *denoted by $\mathbf{H}_\infty(X)$, is defined as*

$$\mathbf{H}_\infty(X) = -\log\left(\max_{x\in\mathcal{X}} X(x)\right).$$

▶ **Definition 2.3** ((n,k)-source)**.** *A distribution X supported on $\{0,1\}^n$ is said to be an* (n,k)-source *if $\mathbf{H}_\infty(X) \geq k$. An (n,k)-source is said to be* flat *if it is uniformly distributed over a subset of $\{0,1\}^n$ of size 2^k.*

The several notions of extractors that we focus on in this work were already covered in Definitions 1.1, 1.2, and 1.8 in Section 1.

Later on, we will exploit the well-known (and not difficult to prove) fact that the Chernoff bound is tight (up to constants in the exponent).

▶ **Lemma 2.4** (Inverse Chernoff bound, see, e.g., [14, Lemma 4, Part 1 with $p = 1/2$])**.** *Suppose $\gamma, D > 0$ are such that $\gamma \leq 1/2$ and $\gamma^2 D \geq 6$, and let Z denote a binomial distribution with D trials and success probability $1/2$. Then,*

$$2^{-D} \cdot \sum_{i=0}^{(1-\gamma)D/2} \binom{D}{i} = \Pr\left[Z \leq \frac{(1-\gamma)D}{2}\right] \geq \exp\left(-\frac{9\gamma^2 D}{2}\right).$$

2.3 Basic boolean functional analysis

In this section, we briefly discuss basic notions from the analysis of boolean functions that we will use later on.

Given a set $S \subseteq [n]$, the Fourier character $\chi_S : \{0,1\}^n \to \{-1,1\}$ is defined as

$$\chi_S(x) = (-1)^{\langle x,s\rangle},$$

where s is the characteristic vector of S (i.e., $s_i = 1$ if and only if $i \in S$). The characters χ_S satisfy $\chi_S(x+y) = \chi_S(x) \cdot \chi_S(y)$ and form an orthonormal basis of the space of functions $f : \{0,1\}^n \to \mathbb{R}$. Consequently, every such function f has a unique Fourier expansion

$$f(x) = \sum_{S \subseteq [n]} \widehat{f}(S) \cdot \chi_S(x),$$

where $\widehat{f}(S) = \mathbb{E}_{x \leftarrow \{0,1\}^n}[f(x) \cdot \chi_S(x)]$ is the Fourier coefficient of f on S.

3 A lower bound for probable extractors

In this section, we follow the high-level approach described in Section 1.3.1 to prove Theorem 1.3, which we restate here for convenience. By the discussion in Section 1.3.2, this result immediately implies the more general Theorem 1.9 for probable extractors.

▶ **Theorem 1.3.** *Every (k,ε)-somewhere extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ must have*

$$D \geq \frac{\ln 2}{2} \cdot \frac{n-k}{\varepsilon}. \tag{4}$$

Fix a (k,ε)-somewhere extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$, and let $A = \mathsf{Ext}(U_n)$. For $z \in \{0,1\}^D$, consider the (unnormalized) distribution P_z defined as

$$P_z(a) = A(a) \cdot \prod_{i=1}^{D} [1 + (-1)^{a_i + z_i}\gamma] = A(a) \sum_{S \subseteq [D]} \chi_S(a+z)\gamma^{|S|} \tag{8}$$

for $a \in \{0,1\}^D$, where $\gamma = (g/D)\ln 2$ and $g = n - k$ is the min-entropy gap. The second equality in (8) holds because for every $S \subseteq [D]$ we have

$$\mathop{\mathbb{E}}_{a\leftarrow\{0,1\}^D}\left[\prod_{i=1}^{D}[1+(-1)^{a_i+z_i}\gamma]\cdot\chi_S(a)\right]=\prod_{i=1}^{D}\mathop{\mathbb{E}}_{a_i\leftarrow\{0,1\}}[(1+(-1)^{a_i+z_i}\gamma)\cdot(-1)^{a_i\cdot S_i}]$$
$$=\prod_{i\in S}\gamma(-1)^{z_i}$$
$$=\gamma^{|S|}\cdot\chi_S(z).$$

Observe that $0<\gamma<1$ since we know $D>g>0$. To see this, note that, if $D\le g$, a simple averaging argument guarantees there is $a\in\{0,1\}^D$ with $|\mathsf{Ext}^{-1}(a)|\ge 2^k$. This implies there is an (n,k)-source X such that $\mathsf{Ext}(X)$ is constant. We can then define the normalizing constant

$$C_z=\sum_{a\in\{0,1\}^D}P_z(a)>0,$$

and we set

$$\overline{P}_z(a)=\frac{P_z(a)}{C_z}.$$

We will fix $z^\star$ to be a choice of z that maximizes C_z, and we let

$$P=\overline{P}_{z^\star}.$$

Note that P is a distribution over $\{0,1\}^D$. We denote the distribution of its i-th coordinate by P_i.

3.1 Proof of Theorem 1.3

Our goal now is twofold: First, we must ensure that

$$P(a)\le 2^g A(a)\quad\forall a\in\{0,1\}^D. \tag{9}$$

Second, we wish to show that

$$\min_{i\in[D]}\Delta(P_i;U_1)\ge\gamma/2. \tag{10}$$

Since Ext is a (k,ε)-somewhere extractor, from (9) and (10) it follows that $\varepsilon\ge\gamma/2$. By the choice of γ above this immediately implies Theorem 1.3.

▶ **Lemma 3.1.** *Condition* (9) *holds for the choice of* $z^\star$ *and* γ *above.*

Proof. First, since $z^\star=\arg\max_z C_z$, we have

$$C_{z^\star}\ge\mathop{\mathbb{E}}_{z\leftarrow\{0,1\}^D}[C_z]=\sum_{a\in\{0,1\}^D}\mathop{\mathbb{E}}_{z\leftarrow\{0,1\}^D}[P_z(a)]=\sum_{a\in\{0,1\}^D}A(a)=1.$$

Therefore, it suffices to show that

$$\prod_{i=1}^{D}[1+(-1)^{z_i^\star+a_i}\gamma]\le 2^g.$$

for every $a\in\{0,1\}^D$. This follows immediately from the fact that

$$(1+\gamma)^D=\left(1+\frac{g\ln 2}{D}\right)^D\le\exp(g\ln 2)=2^g.$$ ◀

▶ **Lemma 3.2.** *We have*

$$\min_{i\in[D]} \Delta(P_i; U_1) \geq \gamma/2.$$

Proof. In order to show the desired inequality, it suffices to prove that

$$\left| \mathop{\mathbb{E}}_{a\sim P}[\chi_{\{i\}}(a)] \right| \geq \gamma \tag{11}$$

for every $i \in [D]$. For any $T \subseteq [D]$ and $z \in \{0,1\}^D$, we have

$$\begin{aligned} C_z \cdot \mathop{\mathbb{E}}_{a\sim P_z}[\chi_T(a)] &= \sum_{a\in\{0,1\}^D} P_z(a)\cdot\chi_T(a) \\ &= \sum_{a\in\{0,1\}^D} \left[A(a) \sum_{S\subseteq[D]} \chi_S(a+z)\gamma^{|S|} \right] \cdot \chi_T(a) \\ &= \sum_{S\subseteq[D]} \chi_S(z)\gamma^{|S|} \sum_{a\in\{0,1\}^D} A(a)\cdot\chi_S(a)\cdot\chi_T(a) \\ &= 2^D \sum_{S\subseteq[D]} \chi_S(z)\gamma^{|S|}\cdot\widehat{A}(S+T), \end{aligned} \tag{12}$$

where the second equality follows from (8), the third equality is true because $\chi_S(a+z) = \chi_S(a)\cdot\chi_S(z)$, and the last equality holds since $\chi_S(a)\cdot\chi_T(a) = \chi_{S+T}(a)$ and by the definition of $\widehat{A}(S+T)$.

For $i \in [D]$ and $b \in \{0,1\}$, define

$$a_b = 2^D \sum_{S\subseteq[D]:S_i=b} \chi_S(z^\star)\gamma^{|S|}\cdot\widehat{A}(S).$$

By setting $T = \emptyset$ and $z = z^\star$, from (12) we obtain

$$C_{z^\star} = C_{z^\star}\cdot \mathop{\mathbb{E}}_{a\sim P}[\chi_\emptyset(a)] = a_0 + a_1. \tag{13}$$

Moreover, setting $T = \{i\}$ and $z = z^\star$ in (12) leads to

$$\begin{aligned} C_{z^\star}\cdot \mathop{\mathbb{E}}_{a\sim P}[\chi_{\{i\}}(a)] &= 2^D \sum_{S\subseteq[D]} \chi_S(z^\star)\gamma^{|S|}\cdot\widehat{A}(S+\{i\}) \\ &= 2^D \sum_{S':=S+\{i\}\subseteq[D]} \chi_{S'+\{i\}}(z^\star)\gamma^{|S'+\{i\}|}\cdot\widehat{A}(S') \\ &= \chi_{\{i\}}(z^\star)\cdot\gamma\cdot 2^D \sum_{S'\subseteq[D]:S'_i=0} \chi_{S'}(z^\star)\cdot\gamma^{|S'|}\widehat{A}(S') \\ &\quad + \frac{\chi_{\{i\}}(z^\star)}{\gamma}\cdot 2^D \sum_{S'\subseteq[D]:S'_i=1} \chi_{S'}(z^\star)\cdot\gamma^{|S'|}\widehat{A}(S') \\ &= \chi_{\{i\}}(z^\star)\left(a_0\gamma + \frac{a_1}{\gamma}\right). \end{aligned} \tag{14}$$

Combining (14) with (13) implies that

$$\left| \mathop{\mathbb{E}}_{a\sim P}[\chi_{\{i\}}(a)] \right| = \left| \frac{a_0\gamma + a_1/\gamma}{a_0 + a_1} \right|. \tag{15}$$

To conclude the proof, we show that $a_1 \geq 0$. Coupled with (15), this yields (11) because then we have

$$|a_0\gamma + a_1/\gamma| \geq \gamma(a_0 + a_1) = \gamma|a_0 + a_1|,$$

where the inequality follows from $a_1 \geq 0$ and the fact that $0 < \gamma < 1$ (recall that $D > g > 0$), and the equality holds because $a_0 + a_1 = C_{z^\star} > 0$.

It remains to show that $a_1 \geq 0$. Let $e_i \in \{0,1\}^D$ be the vector that is 1 at i and 0 elsewhere, and set $z' = z^\star + e_i$. Then, by (12) with $T = \emptyset$ and $z = z' = z^\star + e_i$ we have

$$\begin{aligned} C_{z'} &= 2^D \sum_{b\in\{0,1\}} \sum_{S\subseteq[D]:S_i=b} \chi_S(z^\star + e_i)\gamma^{|S|} \cdot \widehat{A}(S) \\ &= 2^D \sum_{b\in\{0,1\}} (-1)^b \sum_{S\subseteq[D]:S_i=b} \chi_S(z^\star)\gamma^{|S|} \cdot \widehat{A}(S) \\ &= a_0 - a_1, \end{aligned}$$

where the second equality follows from the multiplicative property of χ_S and the fact that $\chi_S(e_i) = (-1)^{S_i}$. Since $z^\star = \arg\max_z C_z$, we conclude that $a_0 + a_1 = C_{z^\star} \geq C_{z'} = a_0 - a_1$, and thus $a_1 \geq 0$. ◀

In Appendix A, we present an alternative proof of Lemma 3.2 which was suggested to us by an anonymous reviewer.

4 Matching upper bound for somewhere extractors

In this section, we prove Theorem 1.4, which we restate here.

▶ **Theorem 1.4.** *For every $\varepsilon \geq \frac{1}{2(1+2^k)}$, there exists a (k,ε)-somewhere extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ with*

$$D = \frac{2^{n-k-1}}{\varepsilon} + 1.$$

Combining this result with Theorem 1.3 in the high min-entropy regime (i.e., $n-k = O(1)$) immediately leads to the following corollary.

▶ **Corollary 4.1.** *The minimum number of seeds required for a (k,ε)-somewhere extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ when $n - k = O(1)$ is $D = \Theta(1/\varepsilon)$.*

It is instructive to compare Corollary 4.1 with analogous results for strong extractors and dispersers, since somewhere extractors lie between the two. With respect to dispersers, the optimal number of seeds in the high min-entropy regime is also $\Theta(1/\varepsilon)$ [22]. Moreover, this is achieved by a uniformly random function with high probability. For strong extractors, the optimal number of seeds is $\Theta(1/\varepsilon^2)$, again achieved by a uniformly random function with high probability. Remarkably, by Corollary 4.1 the optimal number of seeds for (1-bit output) somewhere extractors is $\Theta(1/\varepsilon)$, matching the behavior of dispersers. On the other hand, by Theorem 1.6, a uniformly random function requires $D = \Theta(1/\varepsilon^2)$ to be a (k,ε)-somewhere extractor with non-negligible probability, similarly to strong extractors!

We now proceed to define and analyze the relevant (k,ε)-somewhere extractor Ext that proves Theorem 1.4. In this section, it will be useful to identify the set of inputs $\{0,1\}^n$ with the set of integers $[N]$ for $N = 2^n$. For any N and D, we define the function $\mathsf{Ext} : [N] \times \{0,1,\ldots,E\} \to \{0,1\}$ for $E = \frac{2^{n-k-1}}{\varepsilon}$ via the simple expression

$$\mathsf{Ext}(x,i) = \mathsf{sign}[(x+i) \mod 2E]. \tag{16}$$

In (16), we interpret $(x+i) \mod 2E$ as an integer in $\{-E, -E+1, \ldots, E-1\}$ and $\mathsf{sign}(y) = \mathbf{1}_{\{y\geq 0\}}$.

4.1 Proof of Theorem 1.4

In order to prove the desired statement for the choice of Ext above, we need to show that for every (n,k)-source X there is $i = 0,1,\ldots,E$ such that

$$\Delta(\mathsf{Ext}(X,i);U_1) = \frac{1}{2}|\Pr[\mathsf{Ext}(X,i)=1] - \Pr[\mathsf{Ext}(X,i)=0]| < \varepsilon.$$

Fix an arbitrary (n,k)-source X. For each seed $i = 0,1,\ldots,E$, define

$$\Delta_i = \Pr[\mathsf{Ext}(X,i)=1] - \Pr[\mathsf{Ext}(X,i)=0].$$

The Δ_i's satisfy two important properties. First, observe that

$$\Delta_0 = -\Delta_E \tag{17}$$

since

$$\mathsf{sign}(x \mod 2E) = 1 - \mathsf{sign}[(x+E) \mod 2E]$$

for every x. Second, for every $i \in [E]$ it holds that

$$|\Delta_i - \Delta_{i-1}| \leq 2^{-k} \cdot \left\lceil \frac{N}{2E} \right\rceil = 2^{-k} \cdot \lceil \varepsilon 2^k \rceil \leq 4\varepsilon. \tag{18}$$

To see that (18) holds, it suffices to note that (i) there are at most $\lceil \frac{N}{2E} \rceil = \lceil \varepsilon 2^k \rceil$ integers $x \in [N]$ such that $\mathsf{Ext}(x,i-1) = 0$ but $\mathsf{Ext}(x,i) = 1$ and vice-versa, (ii) $X(x) \leq 2^{-k}$ for every integer x, and (iii) we have $\varepsilon \geq \frac{1}{2(1+2^k)}$. Finally, combining (17) and (18) ensures the existence of $i^\star$ such that $|\Delta_{i^\star}| \leq 2\varepsilon$. Therefore, we have

$$\Delta(\mathsf{Ext}(X,i^\star);U_1) \leq \varepsilon.$$

References

1. Divesh Aggarwal, Siyao Guo, Maciej Obremski, João Ribeiro, and Noah Stephens-Davidowitz. Extractor lower bounds, revisited. *Electronic Colloquium on Computational Complexity (ECCC)*, 26(173), 2019. URL: https://eccc.weizmann.ac.il/report/2019/173.
2. Divesh Aggarwal, Maciej Obremski, João Ribeiro, Luisa Siniscalchi, and Ivan Visconti. How to extract useful randomness from unreliable sources. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020*, pages 343–372, Cham, 2020. Springer International Publishing.
3. Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Sampling algorithms: Lower bounds and applications. In *Proceedings of the Thirty-Third Annual ACM Symposium on Theory of Computing*, STOC '01, page 266–275, New York, NY, USA, 2001. Association for Computing Machinery. doi:10.1145/380752.380810.
4. Boaz Barak, Guy Kindler, Ronen Shaltiel, Benny Sudakov, and Avi Wigderson. Simulating independence: New constructions of condensers, Ramsey graphs, dispersers, and extractors. *J. ACM*, 57(4):20:1–20:52, May 2010. doi:10.1145/1734213.1734214.
5. Boaz Barak, Anup Rao, Ronen Shaltiel, and Avi Wigderson. 2-source dispersers for $n^{o(1)}$ entropy, and Ramsey graphs beating the Frankl-Wilson construction. *Annals of Mathematics*, pages 1483–1543, 2012.
6. Avraham Ben-Aroya, Eshan Chattopadhyay, Dean Doron, Xin Li, and Amnon Ta-Shma. A new approach for constructing low-error, two-source extractors. In *Proceedings of the 33rd Computational Complexity Conference*, CCC '18, pages 3:1–3:19, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.CCC.2018.3.

7 Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *Proceedings of the Forty-eighth Annual ACM Symposium on Theory of Computing*, STOC '16, pages 285–298, New York, NY, USA, 2016. ACM. doi:10.1145/2897518.2897547.

8 Eshan Chattopadhyay and David Zuckerman. Explicit two-source extractors and resilient functions. *Annals of Mathematics*, 189(3):653–705, 2019. URL: https://www.jstor.org/stable/10.4007/annals.2019.189.3.1.

9 Gil Cohen. Local correlation breakers and applications to three-source extractors and mergers. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 845–862, October 2015. doi:10.1109/FOCS.2015.57.

10 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, 2013. doi:10.1137/100783704.

11 Zeev Dvir and Ran Raz. Analyzing linear mergers. *Random Structures & Algorithms*, 32(3):334–345, 2008. doi:10.1002/rsa.20193.

12 Zeev Dvir and Amir Shpilka. An improved analysis of linear mergers. *computational complexity*, 16(1):34–59, May 2007. doi:10.1007/s00037-007-0223-z.

13 Zeev Dvir and Avi Wigderson. Kakeya sets, new mergers, and old extractors. *SIAM Journal on Computing*, 40(3):778–792, 2011. doi:10.1137/090748731.

14 Philip Klein and Neal E. Young. On the number of iterations for Dantzig–Wolfe optimization and packing-covering approximation algorithms. *SIAM Journal on Computing*, 44(4):1154–1172, 2015. doi:10.1137/12087222X.

15 Xin Li. Improved constructions of three source extractors. In *2011 IEEE 26th Annual Conference on Computational Complexity*, pages 126–136, June 2011. doi:10.1109/CCC.2011.26.

16 Xin Li. Extractors for a constant number of independent sources with polylogarithmic min-entropy. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science*, pages 100–109, October 2013. doi:10.1109/FOCS.2013.19.

17 Xin Li. New independent source extractors with exponential improvement. In *Proceedings of the Forty-fifth Annual ACM Symposium on Theory of Computing*, STOC '13, pages 783–792, New York, NY, USA, June 2013. ACM. doi:10.1145/2488608.2488708.

18 Xin Li. Three-source extractors for polylogarithmic min-entropy. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 863–882, October 2015. doi:10.1109/FOCS.2015.58.

19 Xin Li. Improved two-source extractors, and affine extractors for polylogarithmic entropy. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 168–177, October 2016. doi:10.1109/FOCS.2016.26.

20 Chi-Jen Lu, Omer Reingold, Salil Vadhan, and Avi Wigderson. Extractors: Optimal up to constant factors. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 602–611, New York, NY, USA, 2003. ACM. doi:10.1145/780542.780630.

21 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.

22 Jaikumar Radhakrishnan and Amnon Ta-Shma. Bounds for dispersers, extractors, and depth-two superconcentrators. *SIAM Journal on Discrete Mathematics*, 13(1):2–24, 2000. doi:10.1137/S0895480197329508.

23 Anup Rao. An exposition of Bourgain's 2-source extractor. *Electronic Colloquium on Computational Complexity (ECCC)*, 14(034), 2007. URL: http://eccc.hpi-web.de/eccc-reports/2007/TR07-034.

24 Anup Rao. Extractors for a constant number of polynomially small min-entropy independent sources. *SIAM Journal on Computing*, 39(1):168–194, 2009. doi:10.1137/060671218.

25 Ran Raz. Extractors with weak random seeds. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 11–20, New York, NY, USA, 2005. ACM. doi:10.1145/1060590.1060593.

26 Amnon Ta-Shma. On extracting randomness from weak random sources (extended abstract). In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, STOC '96, pages 276–285, New York, NY, USA, 1996. ACM. doi:10.1145/237814.237877.

27 David Zuckerman. Randomness-optimal oblivious sampling. *Random Structures & Algorithms*, 11(4):345–367, 1997. doi:10.1002/(SICI)1098-2418(199712)11:4<345::AID-RSA4>3.0.CO;2-Z.

28 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*, STOC '06, pages 681–690, New York, NY, USA, 2006. ACM. doi:10.1145/1132516.1132612.

A Alternative proof of Lemma 3.2

In this section, we present an alternative proof of Lemma 3.2 that was suggested to us by an anonymous reviewer.

Recall that we defined $P = \frac{P_{z^\star}}{C_{z^\star}}$ for $z^\star = \arg\max_z C_z$, where for arbitrary $z \in \{0,1\}^D$ we defined

$$P_z(a) = A(a) \cdot \prod_{i=1}^{D} [1 + (-1)^{z_i + a_i}\gamma]$$

with $A = \mathsf{Ext}(U_n)$, and

$$C_z = \sum_{a \in \{0,1\}^D} P_z(a).$$

Observe that P is a distribution over $\{0,1\}^D$, and we defined P_i as the distribution of its i-th coordinate. Then, our goal is to show that for every $i \in [D]$ we have

$$\Delta(P_i; U_1) \geq \gamma/2.$$

Equivalently, we must show that

$$|P_i(0) - P_i(1)| \geq \gamma \tag{19}$$

for all $i \in [D]$.

Consider an arbitrary $i \in [D]$. Then, we have

$$C_{z^\star} = (1 + (-1)^{z_i^\star}\gamma)S_0 + (1 - (-1)^{z_i^\star}\gamma)S_1$$

for

$$S_b = \sum_{a : a_i = b} A(a) \prod_{j \neq i} [1 + (-1)^{z_j^\star + a_j}], \quad b \in \{0,1\}.$$

We proceed by cases.

- $z_i^\star = 0$: Then, it holds that $S_0 \geq S_1$. Indeed, if $S_0 < S_1$, we claim that $z^\star$ does not maximize C_z, a contradiction. To see this, consider $z_i' = z^\star + e_i$. Then, we would have

$$\begin{aligned} C_{z'} &= (1-\gamma)S_0 + (1+\gamma)S_1 \\ &> (1+\gamma)S_0 + (1-\gamma)S_1 \\ &= C_z. \end{aligned}$$

As a result, we can compute

$$P_i(0) - P_i(1) = \frac{(1+\gamma)S_0 - (1-\gamma)S_1}{(1+\gamma)S_0 + (1-\gamma)S_1} \geq \gamma,$$

which follows by elementary algebra, using the fact that $S_0 \geq S_1$ and $0 < \gamma < 1$. This implies (19), as desired.

- $z_i^\star = 1$: The proof follows analogously, but symmetrically, to the previous case. First, in this case we have $S_1 \geq S_0$. Then, we have

$$P_i(1) - P_i(0) = \frac{(1+\gamma)S_1 - (1-\gamma)S_0}{(1+\gamma)S_1 + (1-\gamma)S_0} \geq \gamma,$$

again by elementary algebra, since $S_1 \geq S_0$ and $0 < \gamma < 1$. This implies (19), which concludes the proof.

B A lower bound for strong extractors

In this section, we prove Theorem 1.7 via the high-level approach in Section 1.3.1. This yields a different, simpler proof of the optimal lower bound on the number of seeds of (k, ε)-strong extractors, originally obtained by Radhakrishnan and Ta-Shma [22]. We restate Theorem 1.7 here for convenience.

▶ **Theorem 1.7.** *For every $n, k, \varepsilon > 0$ satisfying $n - k \geq 39$ and every (k, ε)-strong extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$ it holds that*

$$D \geq \frac{\ln 2}{18} \cdot \frac{n-k}{\varepsilon^2}.$$

Fix a (k, ε)-strong extractor $\mathsf{Ext} : \{0,1\}^n \times [D] \to \{0,1\}$, and let $A = \mathsf{Ext}(U_n)$. For $z \in \{0,1\}^D$ and $t \leq D$, define the (unnormalized) distribution $P_{z,t}$ over $\{0,1\}^D$ as

$$P_{z,t}(a) = A(a) \cdot \mathbf{1}_{\{\|z-a\|_1 \leq t\}}.$$

The associated normalizing factor $C_{z,t}$ is given by

$$C_{z,t} = \sum_{a \in \{0,1\}^D} P_{z,t}(a) = A(B_t(z)),$$

where $B_t(z)$ denotes the Hamming ball of radius t centered at z and $A(B_t(z)) = \sum_{a \in B_t(z)} A(a)$ denotes its measure under A. Provided that $A(B_t(z)) > 0$, we can then define the normalized distribution $\overline{P}_{z,t}$ by

$$\overline{P}_{z,t}(a) = \frac{P_{z,t}(a)}{C_{z,t}}.$$

B.1 Proof of Theorem 1.7

Taking into account Section 1.3.1, in order to prove Theorem 1.7 we will show, via two easy lemmas, that $P = \overline{P}_{z^\star,t^\star}$ for appropriate choices $z^\star$ and $t^\star$ satisfies

$$P(a) \leq 2^g A(a) \quad \forall a \in \{0,1\}^D \tag{20}$$

and

$$\mathop{\mathbb{E}}_{i \leftarrow [D]}[\Delta(P_i; U_1)] = \frac{1}{D} \left\| \mathop{\mathbb{E}}_{a \sim P}[a] - (1/2, \ldots, 1/2) \right\|_1 \geq c\sqrt{g/D}, \tag{21}$$

where $g = n - k$ is the entropy gap, $c = \sqrt{\frac{\ln 2}{18}}$, and P_i denotes the distribution of the i-th coordinate of P. Properties (20) and (21) imply there is an (n, k)-source X such that $\mathsf{Ext}(X) = P$, and so $\varepsilon \geq c\sqrt{g/D}$. This immediately yields the desired lower bound on D.

▶ **Lemma B.1.** *For $c = \sqrt{\frac{\ln 2}{18}}$ and $C = 39$, if $g = n - k \geq C$, there exists $z^\star \in \{0,1\}^D$ such that* (20) *is satisfied for $t^\star = D/2 - c\sqrt{gD}$.*

Proof. Note that (20) is equivalent to

$$A(B_{t^\star}(z^\star)) \geq 2^{-g}.$$

Moreover, a simple averaging argument (based on the fact that every $y \in \{0,1\}^D$ belongs to the same number of Hamming balls) implies there is $z^\star$ such that $A(B_{t^\star}(z^\star)) \geq 2^{-D} \cdot V_{t^\star}$ (recall $V_{t^\star}$ denotes the volume of a Hamming ball of radius $t^\star$). Fix this choice of $z^\star$. From the choice of c and C above, and since $D > g \geq C$, by the inverse Chernoff bound (Lemma 2.4 with $\gamma = 2c\sqrt{g/D}$) we have

$$2^{-D} \cdot V_{t^\star} \geq \exp\left(-18c^2 g\right) = 2^{-g}$$

for $t^\star = D/2 - c\sqrt{gD}$. ◀

By considering the shifted extractor $\overline{\mathsf{Ext}}(x) = \mathsf{Ext}(x) + z^\star$, without loss of generality we can assume that $z^\star = 0$. Then, we have the following result.

▶ **Lemma B.2.** *For $t = D/2 - \alpha$, it holds that*

$$\left\| \mathop{\mathbb{E}}_{a \sim \overline{P}_{0,t}} [a] - (1/2, \ldots, 1/2) \right\|_1 \geq \alpha.$$

Proof. Note that $\mathbb{E}_{a \sim \overline{P}_{0,t}}[a]$ is a convex combination of elements of $\mathsf{supp}(\overline{P}_{0,t})$. Since $\mathsf{supp}(\overline{P}_{0,t}) \subseteq B_t(0)$, it follows that

$$\left\| \mathop{\mathbb{E}}_{a \sim \overline{P}_{0,t}} [a] \right\|_1 \leq t.$$

Moreover, it holds that $\|(1/2, \ldots, 1/2)\|_1 = D/2$. Consequently, by the triangle inequality we have

$$\left\| \mathop{\mathbb{E}}_{a \sim \overline{P}_{0,t}} [a] - (1/2, \ldots, 1/2) \right\|_1 \geq D/2 - t = \alpha.$$ ◀

Combining Lemmas B.1 and B.2 immediately yields (20) and (21).

A Simpler Strong Refutation of Random k-XOR

Kwangjun Ahn
Department of EECS, Massachusetts Institute of Technology, Cambridge, MA, USA
kjahn@mit.edu

Abstract

Strong refutation of random CSPs is a fundamental question in theoretical computer science that has received particular attention due to the long-standing gap between the information-theoretic limit and the computational limit. This gap is recently bridged by Raghavendra, Rao and Schramm where they study sub-exponential algorithms for the regime between the two limits. In this work, we take a simpler approach to their algorithms and analyses.

2012 ACM Subject Classification Theory of computation

Keywords and phrases Strong refutation, Random k-XOR, Spectral method, Trace power method

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.2

Category RANDOM

Funding *Kwangjun Ahn*: Supported by the the NSF CAREER grant (number: 1846088) and the Kwanjeong Educational Foundation scholarship.

Acknowledgements The author thanks Vijay Bhattiprolu for suggesting the idea of re-scaling entries of the matrix representation for simpler refutation algorithm. The author also thanks anonymous reviewers for various comments that indeed help the author improve the presentation.

1 Introduction

Refutation of random instances of constraint satisfaction problems (random CSPs) is one of the central questions in theoretical computer science with numerous applications. Among many predicates (types of constraints), this paper considers the XOR predicate and studies the strong refutation of the corresponding random CSP. In fact, Allen, O'Donnell and Witmer [2] demonstrate that one can use strong refutation algorithms for random XOR to refute random CSPs with other predicates[1]. In particular, we consider:

▶ **Definition 1** (Random k-XOR). *A* ***random k-XOR*** *with probability p (or equivalently, at density pn^{k-1}) refers to a set $\Phi = \{C_S\}$ of k-XOR constraints over n variables $x \in \{\pm 1\}^n$ obtained as per the following procedure:*

1. *First sample each of the n^k possible k-tuples with probability p independently.*
2. *For each sampled $S = (s_1, s_2, \ldots, s_k) \in [n]^k$, include a k-XOR constraint $C_S : \prod_{i=1}^k x_{s_i} = \eta_S$, where η_S is i.i.d. Rademacher random variable.*

For an assignment $x \in \{\pm 1\}^n$, let $P_\Phi(x)$ be the fraction of constraints satisfied by x.

▶ Remark 2. One can alternatively consider a model where we sample subsets of size k instead of k-tuples (there will be $\binom{n}{k}$ possible subsets in total). However, as noted in [2], the precise details of the random model are not relevant to the results to follow. For simplicity, we follow the prior works [2, 7] and consider the above k-tuple model throughout the paper.

[1] For instance, it is demonstrated that one can refute random k-SAT by reducing it to strong refutations of random ℓ-XOR for $\ell = 1, 2, \ldots, k$.

© Kwangjun Ahn;
licensed under Creative Commons License CC-BY

Under this random k-XOR model, we study the strong refutation problem. To motivate the problem, it is a consequence of standard concentration inequalities that when the density is of $\omega(1)$ (i.e., $pn^{k-1} = \omega(1)$), with high probability, no assignment can satisfy more than a $1/2 + o(1)$ fraction of the constraints. Hence, it is a natural algorithmic question to ask whether one can certify such a fact. More specifically, we consider:

▶ **Definition 3** (Strong refutation). *For a quantity $\alpha = \omega(1)$, an algorithm which takes a k-XOR instance and outputs a quantity $\widehat{P_\Phi}$ is said to strongly refute random k-XOR at density α if it satisfies:*

1. *For any k-XOR instance Φ, $P_\Phi(x) \le \widehat{P_\Phi}$ for all assignments $x \in \{\pm 1\}^n$.*
2. *For a random k-XOR instance Φ with density α, $\widehat{P_\Phi} = 1/2 + o(1)$ with high probability.*

However, the question of developing strong refutation algorithms for the density $\omega(1)$ turns out to be rather intractable. More specifically, the best known guarantees are obtained from spectral methods [2, 3] which require the density to be $\widetilde{\Omega}(n^{k/2-1})$. This computational limit of $\widetilde{\Omega}(n^{k/2-1})$ (also known as *spectral threshold*) is significantly larger than the information-theoretic threshold of $\omega(1)$, and this gap has been conjectured to be fundamental.

Recently, to bridge the gap, Raghavendra, Rao and Schramm investigate sub-exponential refutation algorithms below the spectral threshold [7]. Their results constitute a smooth trade-off between the density and the time complexity required for certifying unsatisfiability. More specifically, their algorithm parametrized by d achieves the following performance: For all $\delta \in [0, 1)$, their algorithm with $d = n^\delta$ finds a certificate at density $\widetilde{\Omega}\left(n^{(k/2-1)(1-\delta)}\right)$ in time $\exp(\widetilde{O}(n^\delta))$. At $\delta \approx 0$, their result recovers the polynomial-time strong refutation result at the spectral threshold, while at $\delta \approx 1$, their result recovers an exponential-time strong refutation at the information-theoretic threshold.

This beautiful result, however, relies on an intricate analysis spanning over 20 pages as well as technical complications in algorithm steps, raising a question of whether one can simplify the analysis as well as the algorithm. This work addresses this question as follows:

1. This work simplifies the key technical component of the analysis in [7] (Section 4). More specifically, the spectral norm analysis [7, Theorem 4.4] is significantly simplified in this work relying on more elementary combinatorial arguments.
2. In addition, for even k, this work also simplifies their refutation algorithm by modifying the technical preprocessing step (Section 5). At a high level, the previous work requires $O(d)$ spectral norm computations of the matrix of size $n^{O(d)\times O(d)}$, whereas the approach in this paper only requires a *single* computation.

As a byproduct of our simpler approach, the theoretical guarantee in this paper comes with less technical conditions and enjoys better refutation performances.

2 Preliminary: spectral strong refutation algorithms

To set the stage for our main result, we first briefly review the spectral refutation algorithms in the prior works [5, 2, 3] that achieve the spectral threshold. For illustrative purpose, we focus throughout on the case when k is even; indeed, the odd k case follows similarly modulo some extra "tricks" to reduce it to the even case (see e.g. [2, Appendix A.2] for details).

We first represent the strong refutation problem as the problem of certifying an upper bound on a polynomial.

▶ **Definition 4** (Constraints tensor). *Given a set of constraints Φ consisting of m constraints $C_{S_1}, \dots, C_{S_m}$, the* constraints tensor *of Φ is a n^k tensor $\boldsymbol{T}^\Phi$ defined as $\boldsymbol{T}^\Phi_S = \eta_{S_a}$ if $S = S_a$ for some $a = 1, \dots, m$ and $\boldsymbol{T}^\Phi_S = 0$ otherwise.*

▶ **Definition 5** (Constraints polynomial). *Given a set of constraints* Φ*, the* constraints polynomial *of* Φ *is a* k*-degree homogeneous polynomial* f^Φ *defined as* $f^\Phi(x) := \langle \boldsymbol{T}^\Phi, x^{\otimes k}\rangle$.

Having the above definitions, it is straightforward to verify the following identity:

$$2m \cdot \left(P_\Phi(x) - \frac{1}{2}\right) = \sum_{\ell=1}^{m} \eta_{(i_\ell,j_\ell)} x_{i_\ell} x_{j_\ell} = \langle \boldsymbol{T}^\Phi, x^{\otimes k}\rangle = f^\Phi(x)$$

$$\iff P_\Phi(x) = \frac{1}{2} + \frac{1}{2m} \cdot f^\Phi(x)\,. \tag{1}$$

Having established (1), the strong refutation problem turns into the problem of certifying a good upper bound on the constraints polynomial:

$$\max_{x\in\{\pm 1\}^n} P_\Phi(x) \le \frac{1}{2} + o(1) \iff \max_{x\in\{\pm 1\}^n} f^\Phi(x) = o(m)\,. \tag{2}$$

Now, the key idea of the spectral refutation is to certify an upper bound on the constraints polynomial by first computing a *matrix representation* of the polynomial and then computing the spectral norm of the matrix. We first formally define matrix representations:

▶ **Definition 6** (Matrix representation [1, Section 9]). *We say an* $n^{k/2} \times n^{k/2}$ *matrix* M *is a matrix representation of a degree-*k *homogeneous polynomial* f *if we have* $f(x) = (x^{\otimes k/2})^\top M x^{\otimes k/2}$*. Here and below, we use* $x^{\otimes k/2}$ *to denote its vector flattening*[2].

If we have a matrix representation M of the constraints polynomial f^Φ, one can certify an upper bound by computing the spectral norm of the matrix representation:

$$\max_{x\in\{\pm 1\}^n} f^\Phi(x) = \max_{x\in\{\pm 1\}^n} (x^{\otimes k/2})^\top M x^{\otimes k/2} \le n^{k/2} \|M\|\,, \tag{3}$$

where the inequality follows since $\left\|x^{\otimes k/2}\right\| = \sqrt{n^{k/2}}$.

Having (3), it is now crucial to find a matrix representation that results in a small spectral norm. It turns out that to achieve the spectral threshold, a simple matrix representation suffices. Let us denote by $\boldsymbol{M}^\Phi$ the natural $n^{k/2} \times n^{k/2}$ flattening of the constraints tensor $\boldsymbol{T}^\Phi$. Certainly $\boldsymbol{M}^\Phi$ is a matrix representation, and hence, its symmetrization is also a matrix representation:

▶ **Definition 7** (Symmetric matrix representation). $S^\Phi := \frac{1}{2}[\boldsymbol{M}^\Phi + (\boldsymbol{M}^\Phi)^\top]$.

Indeed, it follows from a standard result in random matrix theory that the symmetric matrix representation S^Φ constructed from random k-XOR has the spectral norm $o(m)$ with high probability as soon as the density is above the spectral threshold, i.e., $\alpha = \widetilde{\Omega}(n^{k/2-1})$ (see e.g. [2, Appendix A.1] for precise details).

Thus far, we present the spectral refutation algorithms in the prior arts that achieve the spectral threshold. Now, we move on to the result due to Raghavendra, Rao and Schramm [7]. It turns out that for strong refutation below the spectral threshold, one needs to rely on a higher-order symmetry. This will be the subject of the next section.

3 Higher-order symmetry for refutation below spectral threshold

In this section, we discuss the approach based on a higher-order symmetry due to Raghavendra, Rao and Schramm [7]. We remark that a similar technique was independently developed by Bhattiprolu, Guruswami and Lee [4] under the context of finding an upper bound certificate of the tensor injective norm.

[2] More formally, we regard $x^{\otimes k/2}$ as a vector of dimension $nk/2$ rather than as a $n^{k/2}$ tensor.

3.1 Higher-order type-symmetric matrix representation

To illustrate the main idea, we first define the types of the entries:

▶ **Definition 8** (Histogram tuples). *Let* $\mathsf{hist}(I)$ *be the* n*-tuple* $(\alpha_1, \ldots, \alpha_n)$ *such that* α_i *is the number of times* i *appears in* I*, i.e., the histogram of the tuple* I*. Let* $\mathsf{hist}(I)! := \prod_{i=1}^{n} (\alpha_i)!$*, where* $0! = 1$ *by convention.*

▶ **Definition 9** (Types of entries). *Given a matrix representation* M *of a constraints polynomial* f^Φ*, we say two entries* $M_{I,J}$ *and* $M_{I',J'}$ *have the same* type *if* $\mathsf{hist}(I) = \mathsf{hist}(I')$ *and* $\mathsf{hist}(J) = \mathsf{hist}(J')$*, i.e., for all* $i \in [n]$*, the number of* i*'s appearing in* I *(resp.* J*) is equal to that in* I' *(resp.* J'*) .*

With this definition, one can easily notice that the entries of the same type corresponds to the coefficient of the same monomial in f^Φ. Now the key idea of [7] is to consider a matrix representation which distributes the coefficient of a monomial in f^Φ *equally* across the corresponding type of entries. It turns out that such a matrix representation has small spectral norm, resulting in a better refutation certificate.

▶ **Definition 10** (Type-symmetric matrix representation). *We say a matrix representation is type-symmetric if the entries of the same type have the same value.*

To maximize the gain from a type-symmetric matrix representation, [7] indeed considers a higher order matrix representation, which amounts to working with $(f^\Phi)^d$ instead of f^Φ for some $d > 1$ at the cost of increased computational complexity. Given a type-symmetric matrix representation $\boldsymbol{R}^{\Phi,d}$ of $(f^\Phi)^d$ (we defer the formal definition to Definition 13), we have $f^\Phi(x)^d = (x^{\otimes kd/2})^\top \boldsymbol{R}^{\Phi,d} x^{\otimes kd/2}$ from which one can conclude

$$\max_{x \in \{\pm 1\}^n} f^\Phi(x) = n^{k/2} \cdot \max_{x \in \frac{1}{\sqrt{n}} \cdot \{\pm 1\}^n} \left[(x^{\otimes kd/2})^\top \boldsymbol{R}^{\Phi,d} x^{\otimes kd/2} \right]^{1/d} \leq n^{k/2} \cdot \left\| \boldsymbol{R}^{\Phi,d} \right\|^{1/d} . \quad (4)$$

However, as mentioned in [7, Section 4], it turns out that the inequality in (4) is not tight enough for the desired result. To overcome this issue, [7] suggested the technique of removing high multiplicity rows/columns. This will be the subject of the next subsection.

3.2 Overcoming challenge with trimming rows/columns

Before getting into the technique in [7], let us first discuss why the inequality in (4) is not tight. The main reason for the looseness is the fact that the left hand side of the inequality is the maximum over the specific unit vectors of the form $\frac{1}{\sqrt{n}} \cdot \{\pm 1\}^n$, while the spectral norm certificate finds the maximum over all unit vectors. In particular, if the maximum of the spectral norm is achieved by a sparse vector, this certificate would no longer provide a good upper bound.

To cope with this issue, [7] employs the *trimming* step, in which they remove rows and columns of $\boldsymbol{R}^{\Phi,d}$ corresponding to index tuples with high multiplicities, i.e., I's such that coordinate values of $\mathsf{hist}(I)$ are large. This technical step indeed results in a better spectral norm bound as we shall see in Section 4.2.

3.3 Technical challenge of the approach in Raghavendra-Rao-Schramm

However, it turns out that analyzing this higher-order method with the trimming step is rather technical:

1. Note that the construction of symmetric matrix representation results in a rather complicated dependency structure across entries, making it hard to analyze its spectral norm. Indeed, the spectral norm analysis [7, Theorem 4.4] constitutes the main technical component of the analysis in [7].
2. Moreover, it turns out that justifying the validity of the trimming step also requires some technical modification of the algorithm together with an additional careful analysis. At a high level, these complications arise due to the fact that the trimmed matrix is no longer a matrix representation of the constraints polynomial. In particular, their approach requires computations of $O(d)$ spectral norms of matrices of size $n^{O(d)\times O(d)}$.

We will address the above challenges in order in the subsequent sections.

4 A simpler spectral norm analysis

In this section, we provide a simpler spectral norm analysis of the symmetric matrix representation. As we mentioned in the previous section, the symmetric matrix representation has an intricate dependency structure between entries and hence the standard tools such as matrix Chernoff bound [8] does not apply. Hence, we need to rely on more direct analysis based on the trace power method:

▶ **Proposition 11** (Trace power method). *Let $n, \ell \in \mathbb{N}$, let $c \in \mathbb{R}$, and let M be a symmetric $n \times n$ random matrix. Then,*

$$\mathbb{E}\operatorname{Tr}(M^{2\ell}) \le \beta \implies \Pr\left(\|M\| \ge c \cdot \beta^{1/2\ell}\right) \ge 1 - c^{-2\ell}.$$

Proof. The proof follows from the fact that $\|M\|^{2\ell} = \left\|M^{2\ell}\right\| \le \operatorname{Tr}(M^{2\ell})$ together with Markov's inequality. ◀

Hence, to come up with a probabilistic upper bound on the spectral norm, one need to bound the trace power term. However, in contrast to well-known settings in random matrix theory, our matrix of interest M has $kd/2$-tuples for its row/column indices, which renders computing the trace power term more complicated. In particular, for an integer ℓ, the trace power term can be represented as

$$\sum_{I^{(1)},\dots,I^{(2\ell)} \in [n]^{kd/2}} \mathbb{E}\left[\prod_{j=1}^{2\ell} M_{I^{(j)},I^{(j+1)}}\right], \tag{5}$$

where indices are read modulo-2ℓ, i.e., $I_{2\ell+1}$ denotes I_1. As a warm-up, we first analyze the symmetric matrix representation, i.e. $M = \boldsymbol{R}^{\Phi,d}$.

4.1 Warm-up: analysis for higher-order type-symmetric matrix

In this section we apply the trace power method to $M = \boldsymbol{R}^{\Phi,d}$ as a warm-up. Let us first formally define $\boldsymbol{R}^{\Phi,d}$. To that end, we first recall the symmetric matrix representation S^{Φ}. By its definition (Definition 7), S^{Φ} is a $n^{k/2} \times n^{k/2}$ symmetric random matrix with independent mean-zero entries taking values in $[-1, 1]$. Now, let $S^{\Phi,d}$ be the d-th Kronecker power of S^{Φ}, i.e., for $k/2$-tuples $U_1, \dots, U_d$ and $V_1, \dots, V_d$,

$$S^{\Phi,d}_{(U_1,\dots,U_d),(V_1,\dots,V_d)} = S^{\Phi}_{U_1,V_1} \times S^{\Phi}_{U_2,V_2} \times \cdots \times S^{\Phi}_{U_d,V_d}. \tag{6}$$

Now, the symmetric matrix representation is obtained from $S^{\Phi,d}$ by replacing each entry with the average of the entries of the same type as the corresponding entry. To formally define, we begin with some notations:

▶ **Definition 12** (Permutations). *For each positive integers n, q and $I = (i_1, \dots, i_q) \in [n]^q$, let $\mathbb{S}_q$ be the set of permutations on $[q]$. For a permutation $\pi \in \mathbb{S}_q$ and a subtuple $U = (i_{j_1}, \dots, i_{j_\ell})$ of I, let $\pi(U) := (i_{\pi(j_1)}, \dots, i_{\pi(j_\ell)})$.*

Now based on these notations, we formally define $\boldsymbol{R}^{\Phi,d}$ as follows:

▶ **Definition 13** (Higher-order symmetric matrix representation). *For an even integer k and $d \geq 1$, $\boldsymbol{R}^{\Phi,d}$ is an $n^{kd/2 \times kd/2}$ matrix representation of $(f^\Phi)^d$ defined as*

$$\boldsymbol{R}^{\Phi,d}_{I,J} = \frac{1}{|\mathbb{S}_{kd/2}|^2} \sum_{\pi,\sigma \in \mathbb{S}_{kd/2}} S^{\Phi,d}_{\pi(I),\sigma(J)} \,. \tag{7}$$

Now having the formal definition of $\boldsymbol{R}^{\Phi,d}$, one can write the trace power term (5) as follows (where we write each $kd/2$-tuple as $I^{(\cdot)} = (U_1^{(\cdot)}, U_2^{(\cdot)}, \dots, U_d^{(\cdot)})$):

$$\begin{aligned} &\frac{1}{|\mathbb{S}_{kd/2}|^{4\ell}} \cdot \sum_{\substack{I^{(j)} \in [n]^{kd/2} \\ j=1,\dots,2\ell}} \sum_{\substack{\pi_j,\sigma_j \in \mathbb{S}_{kd/2} \\ j=1,\dots,2\ell}} \mathbb{E}\left[\prod_{j=1}^{2\ell} S^{\Phi,d}_{\pi_j(I^{(j)}),\sigma_j(I^{(j+1)})}\right] \\ =&\frac{1}{|\mathbb{S}_{kd/2}|^{4\ell}} \cdot \sum_{\substack{I^{(j)} \in [n]^{kd/2} \\ j=1,\dots,2\ell}} \sum_{\substack{\pi_j,\sigma_j \in \mathbb{S}_{kd/2} \\ j=1,\dots,2\ell}} \mathbb{E}\left[\prod_{j=1}^{2\ell} \prod_{s=1}^{d} S^{\Phi}_{\pi_j(U_s^{(j)}),\sigma_j(U_s^{(j+1)})}\right], \end{aligned} \tag{8}$$

Although (8) looks quite complicated, note that one can actually simplify it further.

▶ **Definition 14** (Partition of the index set according equality). *Given $\{I^{(j)}\}$, $\{\pi_j\}$ and $\{\sigma_j\}$ $(j = 1, \dots, 2\ell)$, we define $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})$ to be the partition of the index set $\mathcal{I} := \{(j, s) \,:\, j = 1, \dots, 2\ell, s = 1, \dots, d\}$ according to the equivalence relation $(j, s) \sim (j', s')$ if and only if $(\pi_j(U_s^{(j)}), \sigma_j(U_s^{(j+1)})) = (\pi_{j'}(U_{s'}^{(j')}), \sigma_{j'}(U_{s'}^{(j'+1)}))$ or $(\pi_j(U_s^{(j)}), \sigma_j(U_s^{(j+1)})) = (\sigma_{j'}(U_{s'}^{(j'+1)}), \pi_{j'}(U_{s'}^{(j')}))$. We denote by $|\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})|$ the number of equivalence classes in the partition.*

Since S^Φ is a symmetric random matrix with mean zero entries, it follows that the summand in (8) corresponding to $\{I^{(j)}\}$, $\{\pi_j\}$ and $\{\sigma_j\}$ is equal to zero if the partition $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})$ contains an equivalence class of odd size.

Hence, in order to have a nonzero summand, every equivalence class of the partition $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})$ must have even size.

▶ **Definition 15.** *Given $\{I^{(j)}\}$, $\{\pi_j\}$ and $\{\sigma_j\}$ $(j = 1, \dots, 2\ell)$, we say the partition of the index set $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})$ is called an* even *partition if all of its equivalence classes have even size.*

When $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})$ is even, since each entry of S^Φ is in $[-1, 1]$, one can easily upper bound the summand of (8) explicitly in terms of the number of equivalence classes in $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\})$:

$$\mathbb{E}\left[\prod_{j=1}^{2\ell} \prod_{s=1}^{d} S^{\Phi}_{\pi_j(U_s^{(j)}),\sigma_j(U_s^{(j+1)})}\right] \leq p^{|\mathsf{Par}(\{I^{(j)}\},\{\pi_j\},\{\sigma_j\})|} \,. \tag{9}$$

Using the upper bound (9), and grouping the trace power term so that each group contains the summand corresponding to the same partition, we obtain

$$(8) \le \frac{1}{\left|\mathbb{S}_{kd/2}\right|^{4\ell}} \sum_{\mathcal{Q}:\ \text{even}} \left[p^{|\mathcal{Q}|} \cdot \mathsf{Num}(\mathcal{Q})\right], \tag{10}$$

where $\mathsf{Num}(\mathcal{Q}) := \left|\left\{(\{I^{(j)}\},\{\pi_j\},\{\sigma_j\}) \ :\ \mathsf{Par}(\{I^{(j)}\},\{\pi_j\},\{\sigma_j\}) = \mathcal{Q}\right\}\right|$. Therefore, to upper bound the trace power term, one needs to upper-estimate $\mathsf{Num}(\mathcal{Q})$ for each $\mathcal{Q}$. Although the counting $\mathsf{Num}(\mathcal{Q})$ looks complicated, the symmetry saves the day.

▶ **Definition 16.** *Num*$(\mathcal{Q} \mid \{\pi_j\}) := \left|\left\{(\{I^{(j)}\},\{\sigma_j\}) \ :\ \mathsf{Par}(\{I^{(j)}\},\{\pi_j\},\{\sigma_j\}) = \mathcal{Q}\right\}\right|$.

First, one can easily verify the following based on a simple symmetry argument (here Id denotes the identity permutation in $\mathbb{S}_{kd/2}$):

▷ Claim 17. $\mathsf{Num}\left(\mathcal{Q} \mid \{\mathsf{Id}\}\right) = \mathsf{Num}\left(\mathcal{Q} \mid \{\pi_j\}\right)$ for any $\{\pi_j\}$.

Proof. See Section A.1. ◁

Due to Claim 17, it follows that:

$$\mathsf{Num}\left(\mathcal{Q}\right) = \left|\mathbb{S}_{kd/2}\right|^{2\ell} \cdot \mathsf{Num}\left(\mathcal{Q} \mid \{\mathsf{Id}\}\right). \tag{11}$$

Hence, with this argument, we reduce the problem of counting triples $(\{I^{(j)}\},\{\pi_j\},\{\sigma_j\})$ into the problem of counting pairs $(\{I^{(j)}\},\{\sigma_j\})$. Now let us further reduce the problem. To that end, we first define:

▶ **Definition 18.** *We say a collection of index tuples* $\{I^{(j)}\}$ *is* $\mathcal{Q}$-valid *if there exist* $\{\sigma_j\}$ *such that* $\mathsf{Par}(\{I^{(j)}\},\{\mathsf{Id}\},\{\sigma_j\})$ *is equal to* $\mathcal{Q}$.

▷ Claim 19. For any $\mathcal{Q}$-valid $\{I^{(j)}\}$, there are at most $(kd/2)^{k|\mathcal{Q}|/2} \cdot \prod_{j=1}^{2\ell} \mathsf{hist}(I^{(j)})!$ different $\{\sigma_j\}$'s such that $\mathsf{Par}\left(\{I^{(j)}\},\{\mathsf{Id}\},\{\sigma_j\}\right) = \mathcal{Q}$.

Proof. The proof is based on an elementary counting argument. See Section A.2. ◁

Due to Claim 19, now we have:

$$\mathsf{Num}\left(\mathcal{Q}\right) = \left|\mathbb{S}_{kd/2}\right|^{2\ell} \cdot (kd/2)^{k|\mathcal{Q}|/2} \cdot \sum_{\{I^{(j)}\}\ :\ \mathcal{Q}\text{-valid}} \left[\prod_{j=1}^{2\ell} \mathsf{hist}(I^{(j)})!\right]. \tag{12}$$

Putting this back to (10), we obtain the following result:

▶ **Theorem 20.** *For even* k *and* $d \ge 1$*, let* $\boldsymbol{R}^{\Phi,d}$ *be the* $n^{kd/2 \times kd/2}$ *higher-order symmetric matrix representation (Definition 13) of random* k*-XOR. Then, the following upper bound on the trace power term holds:*

$$\mathbb{E}\,\mathrm{Tr}((\boldsymbol{R}^{\Phi,d})^{2\ell}) \le \frac{1}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{\mathcal{Q}:\ even} \left[\left(p(kd/2)^{k/2}\right)^{|\mathcal{Q}|} \cdot \sum_{\substack{\{I^{(j)}\}\ :\\ \mathcal{Q}\text{-}valid}} \left[\prod_{j=1}^{2\ell} \mathsf{hist}(I^{(j)})!\right]\right].$$

Having established Theorem 20, one can slightly modify the proof to handle the trimmed matrix from Section 3.2. This will be the focus of the next subsection.

4.2 A simpler spectral norm analysis of the trimmed matrix from Raghavendra-Rao-Schramm

Having established Theorem 20, which explicitly characterizes the upper bound on the trace power term in terms of $\mathsf{hist}(I^{(j)})!$'s, one can now quantitatively understand the trimming technique due to Raghavendra, Rao and Schramm [7]. In particular, we will shortly demonstrate that our Theorem 20 recovers the main technical result [7, Theorem 4.4]. This is remarkable as our proof is much simpler than the original proof in [7].

The problem with the upper bound in Theorem 20 is that the value $\mathsf{hist}(I^{(j)})!$ could be in general large. For instance, if $I^{(j)}$ is the $kd/2$-tuple consisting only of index 1, then $\mathsf{hist}(I^{(j)})! = (kd/2)!$, which turns out to be too large for our desired result. Now having observed this, one can now see how the trimming preprocessing of [7] helps reduce the spectral norm: by removing rows/columns corresponding to the index tuples with high multiplicities, one can significantly reduce the upper bound. More formally, following [7], if we remove the rows/columns corresponding to the index tuples I's such that $\mathsf{hist}(I)$ has a coordinate value larger than $10 \log n$, we have the following:

▶ **Corollary 21.** *For even k and $d \geq 1$, let $\boldsymbol{R}^{\Phi,d,\mathrm{trim}}$ be the $n^{kd/2 \times kd/2}$ matrix obtained from the $\boldsymbol{R}^{\Phi,d}$ (Definition 13) by removing all rows/columns I's such that $\mathsf{hist}(I)$ has a coordinate value larger than $10 \log n$. Assume that $d^{k/2-1} n^{k/2} p > 1$. Then, the following spectral norm bound holds with probability at least $1 - n^{-2}$:*

$$\left\| \boldsymbol{R}^{\Phi,d,\mathrm{trim}} \right\|^{1/d} \leq c \cdot \frac{e^{3k/4} \cdot 10^{5k/2}}{(k/2)^{k/4}} \cdot \frac{n^{k/4} p^{1/2}}{d^{(k-2)/4}} \cdot \log^{5k/2+1} n \,.$$

for some absolute constant $c > 0$.

▶ Remark 22. Although we focus on the even k case throughout the proof for simplicity, we note that a similar argument applies to the case of odd k following the "tricks" [7, Section 4.2] based on Cauchy Schwartz inequality. Consequently, our proof technique provides a simpler proof of the main technical statement for the odd k case [7, Theorem 4.13]

Proof of Corollary 21. From Theorem 20, we have the following upper bound on the trace power term:

$$\mathbb{E}\,\mathrm{Tr}((\boldsymbol{R}^{\Phi,d,\mathrm{trim}})^{2\ell}) \leq \frac{1}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{\mathcal{Q}\colon \text{ even}} \left[\left(p(kd/2)^{k/2} \right)^{|\mathcal{Q}|} \cdot \sum_{\substack{\{I^{(j)}\} \,:\\ \mathcal{Q}\text{-valid}}} \left[\prod_{j=1}^{2\ell} \mathsf{hist}(I^{(j)})! \right] \right] .$$

On the other hand, due to the trimming procedure, each coordinate value of the tuple $\mathsf{hist}(I^{(j)})$ is upper bounded by $10 \log n$, from which we have the following upper bound on the $\mathsf{hist}(I^{(j)})!$:

$$\mathsf{hist}(I^{(j)})! \leq ((10 \log n)!)^{kd/2} \leq (10 \log n)^{5kd \log n} \leq n^{5kd \log(10 \log n)} \,.$$

The trimming step gives us an uniform upper bound on $\mathsf{hist}(I^{(j)})!$, and hence, it suffices to upper bound the number of $\mathcal{Q}$-valid $\{I^{(j)}\}$'s:

▷ Claim 23. For any even partition $\mathcal{Q}$, there are at most $n^{k(|\mathcal{Q}|+d)/2}$ $\mathcal{Q}$-valid $\{I^{(j)}\}$'s.

Proof. The proof is elementary. See Section A.3. ◁

Due to Claim 23, the upper bound on the trace power term becomes:

$$\mathbb{E}\,\mathrm{Tr}((\boldsymbol{R}^{\Phi,d,\mathrm{trim}})^{2\ell}) \le \frac{n^{5kd\log(10\log n)}}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{\mathcal{Q}:\ \text{even}} \left[\left(p(kd/2)^{k/2}\right)^{|\mathcal{Q}|} \cdot n^{k(|\mathcal{Q}|+d)/2}\right]$$
$$= \frac{n^{5kd\log(10\log n)+kd/2}}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{M=1}^{d\ell} \left[N_M \cdot \left(p(nkd/2)^{k/2}\right)^{M}\right], \tag{13}$$

where N_M is the number of even partitions of size M and we have $M \le d\ell$ in the range of summation since an even partition has size at most $d\ell$. Thus, the last ingredient is to bound the number of even partitions:

▷ Claim 24. $N_M \le \binom{2d\ell}{M} \cdot M^{2d\ell-M}$ for all $1 \le M \le d\ell$.

Proof. The first term in the upper bound accounts for the number of different ways of choosing M representative indices in $\mathcal{I}$, and the second term counts the number of different ways of assigning the other indices to the M representative elements. ◁

Due to Claim 24, the upper bound (13) becomes:

$$\frac{n^{5kd\log(10\log n)+kd/2}}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{M=1}^{d\ell} \left[\binom{2d\ell}{M} \cdot M^{2d\ell-M} \cdot \left(p(nkd/2)^{k/2}\right)^{M}\right]. \tag{14}$$

Having established (14), the rest of the proof is straightforward calculations. We first upper bound each term in the above summand as follows: (i) $\binom{2d\ell}{M} \le 2^{2d\ell}$, (ii) $M^{2d\ell-M} \le (d\ell)^{2d\ell-M} \le d^{2d\ell-M}\ell^{2d\ell}$, and (iii) $\left((kdn/2)^{k/2}p\right)^{M} \le \left((dn)^{k/2}p\right)^{M} \cdot (k/2)^{kd\ell/2}$. Then, the summand in (14) is upper bounded by

$$2^{2d\ell} \cdot d^{2d\ell-M}\ell^{2d\ell} \cdot (k/2)^{kd\ell/2} \left((dn)^{k/2}p\right)^{M} = (2\ell)^{2d\ell}(k/2)^{kd\ell/2} \cdot d^{2d\ell} \left(d^{k/2-1}n^{k/2}p\right)^{M}.$$

Using this upper bound, it follows that

$$(14) \le \frac{(2\ell)^{2d\ell}(k/2)^{kd\ell/2}n^{5kd\log(10\log n)+kd/2}d^{2d\ell}}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \cdot \sum_{M=1}^{d\ell} \left(d^{k/2-1}n^{k/2}p\right)^{M}$$
$$\le \frac{(2\ell)^{2d\ell}(k/2)^{kd\ell/2}e^{kd\ell}n^{5kd\log(10\log n)+kd/2}d^{2d\ell}}{(kd/2)^{kd\ell}} \cdot d\ell \cdot \left(d^{k/2-1}n^{k/2}p\right)^{d\ell}, \tag{15}$$

where the inequality follows from the facts that $|\mathbb{S}_{kd/2}|^{2\ell} = ((kd/2)!)^{2\ell} \ge (\frac{kd/2}{e})^{kd\ell}$ ($\because n! \ge (n/e)^n$) and $d^{k/2-1}n^{k/2}p > 1$. Reorganizing terms in (15), we obtain

$$(2\ell)^{2d\ell+1}e^{kd\ell}(k/2)^{-kd\ell/2}d^{-kd\ell/2+d\ell+1}n^{kd\ell/2+5kd\log(10\log n)+kd/2}p^{d\ell}.$$

Invoking Proposition 11 and using the fact that $f(x) = x^{1/x}$ is bounded on $[1,\infty)$, $\|M\|^{1/d}$ is upper bounded by

$$c \cdot (2\ell)e^{k/2}(k/2)^{-k/4}d^{-k/4+1/2}n^{k/4+5k\log(10\log n)/(2\ell)+k/(4\ell)}p^{1/2}$$

with probability at least $1 - e^{-2\ell}$ for some absolute constant $c > 0$. Choosing $\ell = \log n$, we complete the proof. ◀

Thus far, we have addressed the first challenge in Section 3.3 by developing a simpler spectral norm analysis of the type-symmetric representation as well as the trimmed matrix. Now, we move on to the second challenge: as mentioned in Section 3.3, the trimmed matrix $\boldsymbol{R}^{\Phi,d,\mathrm{trim}}$ is no longer a matrix representation of $(f^{\Phi})^d$, it requires additional non-trivial modifications of the algorithm steps as well as analysis.

5 A simpler spectral refutation with re-scaling entries

In this section, we address the second challenge from Section 3.3 and develop a simpler spectral refutation algorithm. Our main idea is to re-scale the rows/columns of $\boldsymbol{R}^{\Phi,d}$. To describe our re-scaling step, we first revisit the upper bound from Theorem 20:

$$\mathbb{E}\,\mathrm{Tr}((\boldsymbol{R}^{\Phi,d})^{2\ell}) \le \frac{1}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{\mathcal{Q}:\text{ even}} \left[\left(p(kd/2)^{k/2}\right)^{|\mathcal{Q}|} \cdot \sum_{\substack{\{I^{(j)}\}:\\ \mathcal{Q}\text{-valid}}} \left[\prod_{j=1}^{2\ell} \mathsf{hist}(I^{(j)})! \right] \right]. \tag{16}$$

As we have discussed in Section 4.2, we need to cancel out the $\mathsf{hist}(I^{(j)})!$ terms in the bound to reduce the spectral norm. Our approach is to appropriately re-scale $\boldsymbol{R}^{\Phi,d}$ so that one can remove the $\prod_{j=1}^{2\ell}\mathsf{hist}(I^{(j)})!$ terms in the upper bound (16). In particular, if we divide the (I,J)-th entry of $\boldsymbol{R}^{\Phi,d}$ by $\sqrt{\mathsf{hist}(I)!\cdot\mathsf{hist}(J)!}$, the $\prod_{j=1}^{2\ell}\mathsf{hist}(I^{(j)})!$ term will be exactly canceled out by the re-scaling. More formally, we define the following vector and its corresponding diagonal matrix:

▶ **Definition 25** (Re-scaling factors). *Let* hist *be an* $n^{kd/2}$*-dimensional vector whose* I*-th coordinate is defined as* $\mathsf{hist}_I := \sqrt{\mathsf{hist}(I)!}$ *for each* $I \in [n]^{kd/2}$*. We define* D_{hist} *to be an* $n^{kd/2}\times n^{kd/2}$ *diagonal matrix whose* (I,I)*-th entry is defined as* hist_I*.*

Using Definition 25, one can precisely achieve the re-scaling discussed above as follows:

▶ **Definition 26** (Re-scaled matrix representation). $\boldsymbol{R}^{\Phi,d,\mathsf{rescale}} := D_{\mathsf{hist}}^{-1}\cdot\boldsymbol{R}^{\Phi,d}\cdot D_{\mathsf{hist}}^{-1}$.

Then, following the same proof as that of Corollary 21, one can prove the following spectral norm bound:

▶ **Corollary 27.** *For even* k *and* $d\ge 1$*, let* $\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}$ *be the* $n^{kd/2\times kd/2}$ *matrix obtained from the* $\boldsymbol{R}^{\Phi,d}$ *by re-scaling the rows/columns as per* (26)*. Assume that* $d^{k/2-1}n^{k/2}p>1$*. Then, the following spectral norm bound holds with probability at least* $1-n^{-2}$*:*

$$\left\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\right\|^{1/d} \le c\cdot\frac{e^{3k/4}}{(k/2)^{k/4}}\cdot\frac{n^{k/4}p^{1/2}}{d^{(k-2)/4}}\cdot\log n\,.$$

for some absolute constant $c>0$*.*

▶ Remark 28. Note that the spectral norm bound in Corollary 27 is better than the bound due to the trimming step (Corollary 21). This improvement actually leads to a better strong refutation guarantee as we shall see in Theorem 32. Also see Section 6 for an extensive comparison with [7].

Proof. Due to the re-scaling factor, following the proof of Theorem 20, we obtain the following bound on the trace bower term without the $\prod_{j=1}^{2\ell}\mathsf{hist}(I^{(j)})!$ term:

$$\mathbb{E}\,\mathrm{Tr}((\boldsymbol{R}^{\Phi,d,\mathsf{rescale}})^{2\ell}) \le \frac{1}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{\mathcal{Q}:\text{ even}} \left[\left(p(kd/2)^{k/2}\right)^{|\mathcal{Q}|} \cdot \sum_{\substack{\{I^{(j)}\}:\\ \mathcal{Q}\text{-valid}}} [1] \right].$$

Now due to Claims 23 and 24, one can further upper bound the trace power term by

$$\frac{n^{kd/2}}{\left|\mathbb{S}_{kd/2}\right|^{2\ell}} \sum_{M=1}^{d\ell} \left[\binom{2d\ell}{M} \cdot M^{2d\ell-M} \cdot \left(p(nkd/2)^{k/2}\right)^{M} \right], \tag{17}$$

which is better than (14) by a multiplicative factor of $n^{5kd\log(10\log n)}$. Now, following the exact same calculations as in the proof of Corollary 21 and choosing $\ell = \log n$, one can easily notice that the improvement by a multiplicative factor of $n^{5kd\log(10\log n)}$ results in an improvement in the final bound by a multiplicative factor of $n^{5k\log(10\log n)/(2\ell)} = n^{5k\log(10\log n)/(2\log n)} = (10\log n)^{5k/2}$, which completes the proof. ◀

With this re-scaled matrix $\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}$, one can also easily come up with a valid certificate for strong refutation (Definition 3):

▶ **Proposition 29.** *For any k-XOR instance Φ and assignment $x \in \{\pm 1\}^n$, we have*

$$\left|P_\Phi(x) - \frac{1}{2}\right| \le \frac{1}{2m}\left[\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\| \cdot \left(\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)!\right)\right]^{1/d}.$$

In other words, $\frac{1}{2} + \frac{1}{2m}[\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\| \cdot (\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)!)]^{1/d}$ is a valid certificate for strong refutation.

Proof. First, since $\boldsymbol{R}^{\Phi,d}$ is a matrix representation of $(f^\Phi)^d$, we have

$$f^\Phi(x)^d = (x^{\otimes kd/2})^\top \boldsymbol{R}^{\Phi,d} x^{\otimes kd/2}.$$

Hence, it follows that

$$\begin{aligned} f^\Phi(x)^d &= (D_{\mathsf{hist}} x^{\otimes kd/2})^\top \cdot D_{\mathsf{hist}} D_{\mathsf{hist}}^{-1} \cdot \boldsymbol{R}^{\Phi,d} \cdot D_{\mathsf{hist}}^{-1} D_{\mathsf{hist}} \cdot x^{\otimes kd/2} \\ &= (D_{\mathsf{hist}} x^{\otimes kd/2})^\top \cdot \boldsymbol{R}^{\Phi,d,\mathsf{rescale}} \cdot D_{\mathsf{hist}} x^{\otimes kd/2}. \end{aligned}$$

Consequently, we have

$$|f^\Phi(x)^d| \le \left\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\right\| \cdot \left\|D_{\mathsf{hist}} x^{\otimes kd/2}\right\|^2 = \left\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\right\| \cdot \left(\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)!\right),$$

where the equality is due to the fact that $x^{\otimes kd/2}$ is an $n^{kd/2}$-dimensional vector with coordinates equal to ± 1. Therefore, the proposition follows thanks to the identity (1), which reads $P_\Phi(x) = \frac{1}{2} + \frac{1}{2m} \cdot f^\Phi(x)$. ◀

Hence, in order to guarantee that the certificate from Proposition 29 works, our last ingredient is to show that the term $(\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)!)$ is not too large compared to $\left\|x^{\otimes kd/2}\right\|^2 = n^{kd/2}$.

▶ **Proposition 30.** *For even k and $d \ge 1$,*

$$\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)! = \frac{(kd/2+n-1)!}{(n-1)!} = (kd/2+n-1)(kd/2+n-2)\cdots n.$$

In particular, if $d \le n$, we have $\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)! \le (k/2+1)^{kd/2} n^{kd/2}$.

Proof. We first group the terms in the summation according to the value of $\mathsf{hist}(I)$:

$$\sum_{I\in[n]^{kd/2}} \mathsf{hist}(I)! = \sum_{\substack{(s_1,s_2,\ldots,s_n)\in(\mathbb{Z}_{\ge 0})^n: \\ \sum_i s_i = kd/2}} \sum_{\substack{I\in[n]^{kd/2}: \\ \mathsf{hist}(I)=(s_1,s_2,\ldots,s_n)}} \prod_{i=1}^n (s_i)!. \tag{18}$$

For each $(s_1, s_2, \ldots, s_n) \in (\mathbb{Z}_{\geq 0})^n$, there are $\frac{(kd/2)!}{\prod_{i=1}^n (s_i)!}$ different I's such that $\mathsf{hist}(I) = (s_1, s_2, \ldots, s_n)$. Hence, the right hand side of (18) becomes

$$\sum_{\substack{(s_1,s_2,\ldots,s_n)\in(\mathbb{Z}_{\geq 0})^n: \\ \sum_i s_i = kd/2}} (kd/2)! = (kd/2)! \cdot \left| \left\{ (s_1, s_2, \ldots, s_n) \in (\mathbb{Z}_{\geq 0})^n : \sum_i s_i = kd/2 \right\} \right| .$$

It is a simple enumerative combinatorics (c.f. stars and bars argument) to show that the number of feasible $(s_1, \ldots, s_n)$'s is equal to $\binom{kd/2+n-1}{n-1} = \binom{kd/2+n-1}{kd/2}$. Therefore, the summation is equal to

$$\binom{kd/2+n-1}{kd/2} \cdot (kd/2)! = (kd/2+n-1)(kd/2+n-2)\cdots n\,,$$

which completes the proof. ◀

Combining what we have obtained thus far, one can consider the following simpler refutation algorithm based on re-scaling entries:

Algorithm 1 A simpler strong refutation algorithm with parameter d for even k.

Input: A k-XOR instance Φ on n variables consisting of m clauses $C_{S_1}, \ldots, C_{S_m}$ for distinct tuples $S_1, \ldots, S_m \in [n]^k$ and a parameter $d \in \mathbb{N}$.

1. Construct a higher-order symmetric matrix representation $\boldsymbol{R}^{\Phi,d}$ based on the k-XOR instance Φ as per Definition 13.
2. Compute $\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}$ as per Definition 26.

Output: $\widehat{P_\Phi} := \frac{1}{2} + \frac{1}{2m} \left\| \boldsymbol{R}^{\Phi,d,\mathsf{rescale}} \right\|^{1/d} \cdot \left(\frac{(kd/2+n-1)!}{(n-1)!} \right)^{1/d}$.

▶ Remark 31. A similar idea of re-scaling rows/columns with diagonal matrices to obtain a better certificate also appeared in the MAXCUT literature; see e.g. [6, Theorem 2.2].

▶ **Theorem 32.** *Let $d \leq n$ be positive integers and k be an even integer. For any instance Φ of k-XOR, the output $\widehat{P_\Phi}$ of Algorithm 1 satisfies $|P_\Phi(x) - \frac{1}{2}| \leq \widehat{P_\Phi} - \frac{1}{2}$ for any $x \in \{\pm 1\}^n$. Assume further that Φ is an instance of random k-XOR with probability p (Definition 1). If $p \cdot d^{k/2-1} n^{k/2} > 1$, the following bound holds with probability at least $1 - O(n^{-1})$ for some absolute constant $c > 0$:*

$$\widehat{P_\Phi} - \frac{1}{2} \leq c \cdot \frac{\log n}{\sqrt{d^{k/2-1} n^{k/2} p}} \cdot \frac{e^{3k/4} \cdot (k/2+1)^{k/2}}{(k/2)^{k/4}} .$$

In particular, Algorithm 1 with parameter d certifies with high probability that $P_\Phi(x)$ is equal to $1/2 \pm o(1)$ for any $x \in \{\pm 1\}^n$ whenever $n^{k-1} p = \omega((n/d)^{k/2-1} \log^2 n)$.

Proof. First from Proposition 29, we have

$$\left| P_\Phi(x) - \frac{1}{2} \right| \leq \frac{1}{2m} \left[\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\| \cdot \left(\sum_{I \in [n]^{kd/2}} \mathsf{hist}(I)! \right) \right]^{1/d} = \widehat{P_\Phi} - \frac{1}{2}\,, \tag{19}$$

where the equality is due to Proposition 30. Hence the first part of the theorem is proved. As for the second part, it follows from Corollary 27 and Proposition 30 that with probability at least $1 - O(n^{-2})$:

$$\left[\|\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}\| \cdot \left(\sum_{I \in [n]^{kd/2}} \mathsf{hist}(I)! \right) \right]^{1/d} \leq c \cdot \frac{n^{3k/4} p^{1/2}}{d^{(k-2)/4}} \log n \cdot \frac{e^{3k/4} \cdot (k/2+1)^{k/2}}{(k/2)^{k/4}} \tag{20}$$

for some absolute constant $c > 0$. Next, it follows from a standard concentration inequality (e.g. Chernoff bound) that with probability at least (say) $1 - n^{-10}$, $m \geq pn^k/2$. Putting these bounds back to (19), we obtain

$$\begin{aligned} \left| P_\Phi(x) - \frac{1}{2} \right| &\leq \frac{1}{pn^k} \cdot c \cdot \frac{n^{3k/4} p^{1/2}}{d^{(k-2)/4}} \log n \cdot \frac{e^{3k/4} \cdot (k/2+1)^{k/2}}{(k/2)^{k/4}} \\ &= c \cdot \frac{\log n}{\sqrt{d^{k/2-1} n^{k/2} p}} \cdot \frac{e^{3k/4} \cdot (k/2+1)^{k/2}}{(k/2)^{k/4}}, \end{aligned}$$

and hence, the second part of the theorem also follows. ◀

6 Comparison with Raghavendra-Rao-Schramm

We compare Algorithm 1 with the refutation algorithm of Raghavendra, Rao and Schramm [7]. First, the algorithm steps in this paper is simpler than that of [7]. As we have discussed earlier, the trimming step in the algorithm of [7] causes some technical complications as the resulting matrix is no longer a matrix representation of $(f^\Phi)^d$. Indeed, their algorithm first constructs matrices of size $n^{kj/2} \times n^{kj/2}$ for $j \in [\delta d, d]$ and computes the spectral norms of those matrices to design a refutation certificate; see [7, Section 4.1.1] for details. This is in stark contrast with Algorithm 1 which only computes the spectral norm of a *single* matrix $\boldsymbol{R}^{\Phi,d,\mathsf{rescale}}$ of size $n^{kd/2} \times n^{kd/2}$. In addition, while their certificate requires non-trivial analysis [7, Section 4.1.1] to guarantee its validity, the validity of our certificate $\widehat{P_\Phi}$ readily follows as we saw in Proposition 29.

As a result of the simpler approach in this paper, the theoretical guarantee in this paper comes with less technical conditions and enjoys a better refutation guarantee as well as density requirement. More specifically, unlike the guarantee in [7], our main theorem does not require a technical condition like $d \log n = O(n)$. Moreover, the density requirement for strong refutation reads $n^{k-1}p = \omega((n/d)^{k/2-1} \log^{2k} n)$ in [7], which is worse than that of this paper by a poly-logarithmic factor (recall that the requirement in Theorem 32 reads $n^{k-1}p = \omega((n/d)^{k/2-1} \log^2 n)$). Lastly, even when the density requirement is fulfilled, their refutation guarantee reads $\frac{1}{2} + \gamma + o(1)$ for some constant $\gamma > 0$ that depends on a hyperparameter in the trimming step. On the other hand, this constant γ does not appear in the refutation guarantee of this paper.

7 Conclusion

In this paper, we establish a simpler approach to strong refutation of random k-XOR below the spectral threshold. Our simplification is two-fold. First, we provide a simpler spectral norm analysis of the certificate matrix of the previous work [7] (Section 4). Second, we develop a simple strong refutation algorithm for the even k case (Section 5). Thanks to our simpler approach, our main result (Theorem 32) enjoys a better theoretical guarantee under less assumptions. It is important to note that a recent work by Wein, El Alaoui and

Moore also establishes a simpler strong refutation algorithm for random even k-XOR [9, Theorem F.1] with a different approach. Given the successful simplifications for the even k case, it would be interesting to see if one can come up with a simpler strong refutation algorithm for the odd k case.

References

1 Kwangjun Ahn, Dhruv Medarametla, and Aaron Potechin. Graph matrices: Norm bounds and applications. *arXiv*, 2020. `arXiv:1604.03423`.

2 Sarah R Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *Proceedings of the 56th FOCS*, pages 689–708. IEEE, 2015.

3 Boaz Barak and Ankur Moitra. Noisy tensor completion via the sum-of-squares hierarchy. In *COLT*, pages 417–445, 2016.

4 Vijay Bhattiprolu, Venkatesan Guruswami, and Euiwoong Lee. Sum-of-Squares Certificates for Maxima of Random Tensors on the Sphere. In *APPROX/RANDOM 2017*, volume 81, pages 31:1–31:20. LIPIcs, 2017.

5 Amin Coja-Oghlan, Andreas Goerdt, and André Lanka. Strong refutation heuristics for random k-SAT. *Combinatorics, Probability and Computing*, 16(1):5–28, 2007.

6 Charles Delorme and Svatopluk Poljak. The performance of an eigenvalue bound on the max-cut problem in some classes of graphs. *Discrete Mathematics*, 111(1-3):145–156, 1993.

7 Prasad Raghavendra, Satish Rao, and Tselil Schramm. Strongly refuting random CSPs below the spectral threshold. In *Proceedings of the 49th STOC*, pages 121–131. ACM, 2017. `arXiv:1605.00058`.

8 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

9 Alexander S Wein, Ahmed El Alaoui, and Cristopher Moore. The kikuchi hierarchy and tensor pca. In *Proceedings of the 60th FOCS*, pages 1446–1468. IEEE, 2019.

A Deferred proofs of claims

A.1 Proof of Claim 17

Proof. Recall that Claim 17 reads $\mathsf{Num}\left(\mathcal{Q} \mid \{\mathsf{Id}\}\right) = \mathsf{Num}\left(\mathcal{Q} \mid \{\pi_j\}\right)$ for any $\{\pi_j\}$. Let us arbitrarily fix a collection of permutations $\{\pi_j\}$. The main observation is that for any $\{I^{(j)}\}$ and $\{\sigma_j\}$, we have $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\}) = \mathsf{Par}(\{\pi_j(I^{(j)})\}, \{\mathsf{Id}\}, \{\sigma_j \circ \pi_j^{-1}\})$. This is a straightforward consequence of Definition 14. Hence, there is an one-to-one correspondence between the collection of pairs $(\{I^{(j)}\}, \{\sigma_j\})$ such that $\mathsf{Par}(\{I^{(j)}\}, \{\pi_j\}, \{\sigma_j\}) = \mathcal{Q}$ and the collection such that $\mathsf{Par}(\{I^{(j)}\}, \{\mathsf{Id}\}, \{\sigma_j\}) = \mathcal{Q}$. This concluded the proof. ◀

A.2 Proof of Claim 19

Proof. We first restate Claim 19: for any $\mathcal{Q}$-valid$\{I^{(j)}\}$, there are at most $(kd/2)^{k|\mathcal{Q}|/2} \cdot \prod_{j=1}^{2\ell} \mathsf{hist}(I^{(j)})!$ different $\{\sigma_j\}$'s such that $\mathsf{Par}\left(\{I^{(j)}\}, \{\mathsf{Id}\}, \{\sigma_j\}\right) = \mathcal{Q}$.

We bound the number of feasible $\{\sigma_j\}$'s as we go through the index set $\mathcal{I} = \{(j, s) \ : \ j = 1, \ldots, 2\ell, s = 1, \ldots, d\}$ in the lexicographical order, i.e., $(1, 1)$, $(1, 2)$, $\ldots$, $(1, d)$, $(2, 1)$, $\ldots$ and so on. As we read the indices in such an order, we call an index (j, s) *new* if $(U_s^{(j)}, \sigma_j(U_s^{(j+1)}))$ is not equivalent to the previously appeared indices. Consider the indices $(j, 1), (j, 2), \ldots, (j, d)$ for a fixed $j \in [2\ell]$. We consider two different scenarios:

1. First, suppose that all indices $(j, 1), (j, 2), \ldots, (j, d)$ are old. Then it should be the case that for each $m = 1, \ldots, kd/2$, $\sigma_j(m)$ is chosen so that the $\sigma_j(m)$-th coordinate of $I^{(j+1)}$ respects the previous appeared equivalent index. Having observed this, it readily follows that there are $\mathsf{hist}(I^{(j+1)})!$ different choices for $\sigma_j(1), \ldots \sigma_j(kd/2)$ considering the permutation.

2. Now, suppose that there are $d_{\text{new}}^{(j)}$ new indices among $(j,1),(j,2),\dots,(j,d)$. For simplicity, assume that $U_{j,1},\dots,U_{j,d_{\text{new}}^{(j)}}$ are new. Choosing the values $\sigma_j(1),\sigma_j(2),\dots,\sigma_j(kd_{\text{new}}^{(j)}/2)$ arbitrarily, there are at most

$$(kd/2)(kd/2-1)\cdots(kd/2-kd_{\text{new}}^{(j)}/2+1) \le (kd/2)^{kd_{\text{new}}^{(j)}/2}$$

different choices for $\sigma_j(1),\sigma_j(2),\dots,\sigma_j(kd_{\text{new}}/2)$. A similar counting to previous case yields that for the remaining values there are at most $\mathsf{hist}(I^{(j+1)})!$ different choices.

Taking a product over all j's, we complete the proof since $\sum_{m=1}^{2\ell} d_{\text{new}}^{(m)} = |\mathcal{Q}|$. ◀

A.3 Proof of Claim 23

Proof. Let $\mathcal{Q}$ be an even partition. We count the number of possible $\mathcal{Q}$-valid $\{I^{(j)}\}$'s. First, let us choose $I^{(1)}$ arbitrarily. Note that there are $n^{kd/2}$ different ways of choosing I_1. Now, consider $I_2,\dots,I_{2\ell}$. Similar to the proof of Claim 19, we will bound the number of feasible choices s we go through the index set $\mathcal{I} = \{(j,s) \ : \ j=1,\dots,2\ell, s=1,\dots,d\}$ in the lexicographical order. Again, we call an index (j,s) *new* if $(\pi_j(U_s^{(j)}),\sigma_j(U_s^{(j+1)}))$ is not equivalent to the previously appeared indices.

Note that we only need to consider new indices because the tuples of old indices are fully determined by their previous appearance. We begin with the tuples $(1,1),(1,2),\dots,(1,d)$. Whenever we encounter a new tuple, say $(U_s^{(1)},\sigma_1(U_s^{(2)}))$, we only need to specify $\sigma_1(U_s^{(2)})$ since $I^{(1)}$ is already fully specified. Hence, there are at most $n^{kd_{\text{new}}^{(1)}/2}$ different ways of choosing $I^{(2)}$, where $d_{\text{new}}^{(1)}$ is the number of new indices among $(1,1),(1,2),\dots,(1,d)$. By similar arguments, inductively for $j=2,3,\dots,2\ell$, there are at most $n^{kd_{\text{new}}^{(j)}/2}$ different ways of choosing $I^{(j)}$. Taken collectively, we obtain the result since $\sum_{j=1}^{2\ell} d_{\text{new}}^{(j)} = |\mathcal{Q}|$. ◀

Iterated Decomposition of Biased Permutations via New Bounds on the Spectral Gap of Markov Chains

Sarah Miracle
University of St. Thomas, St. Paul, MN, USA
https://www.stthomas.edu/cisc/faculty/sarah-miracle.html
sarah.miracle@stthomas.edu

Amanda Pascoe Streib
Center for Computing Sciences, Bowie, MD, USA
ampasco@super.org

Noah Streib
Center for Computing Sciences, Bowie, MD, USA
nsstrei@super.org

Abstract

In this paper, we address a conjecture of Fill [Fill03] about the spectral gap of a nearest-neighbor transposition Markov chain $\mathcal{M}_{\text{nn}}$ over biased permutations of $[n]$. Suppose we are given a set of input probabilities $\mathcal{P} = \{p_{i,j}\}$ for all $1 \leq i, j \leq n$ with $p_{i,j} = 1 - p_{j,i}$. The Markov chain $\mathcal{M}_{\text{nn}}$ operates by uniformly choosing a pair of adjacent elements, i and j, and putting i ahead of j with probability $p_{i,j}$ and j ahead of i with probability $p_{j,i}$, independent of their current ordering.

We build on previous work [25] that analyzed the spectral gap of $\mathcal{M}_{\text{nn}}$ when the particles in $[n]$ fall into k classes. There, the authors iteratively decomposed $\mathcal{M}_{\text{nn}}$ into simpler chains, but incurred a multiplicative penalty of n^{-2} for each application of the decomposition theorem of [23], leading to an exponentially small lower bound on the gap. We make progress by introducing a new complementary decomposition theorem. We introduce the notion of ϵ-orthogonality, and show that for ϵ-orthogonal chains, the complementary decomposition theorem may be iterated $O(1/\sqrt{\epsilon})$ times while only giving away a constant multiplicative factor on the overall spectral gap. We show the decomposition given in [25] of a related Markov chain $\mathcal{M}_{\text{pp}}$ over k-class particle systems is $1/n^2$-orthogonal when the number of particles in each class is at least $C \log n$, where C is a constant not depending on n. We then apply the complementary decomposition theorem iteratively n times to prove nearly optimal bounds on the spectral gap of $\mathcal{M}_{\text{pp}}$ and to further prove the first inverse-polynomial bound on the spectral gap of $\mathcal{M}_{\text{nn}}$ when k is as large as $\Theta(n/\log n)$. The previous best known bound assumed k was at most a constant.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains

Keywords and phrases Markov chains, Permutations, Decomposition, Spectral Gap, Iterated Decomposition

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.3

Category RANDOM

Related Version A full version of the paper is available at https://arxiv.org/abs/1910.05184.

Acknowledgements We wish to thank Dana Randall for several useful discussions about the decomposition method.



© Sarah Miracle, Amanda Pascoe Streib, and Noah Streib;
licensed under Creative Commons License CC-BY



1 Introduction

For $n \in \mathbb{N}$, the problem of generating permutations of $[n] = \{1, 2 \ldots, n\}$ at random is foundational in the history of computer science [19]. Markov chains for sampling permutations arise in a variety of contexts, including self-organizing lists [17, 30], card shuffling [2, 32], and search engines [5]. The spectral gap of a Markov chain provides a measure of the rate of convergence to stationarity, which is crucial to the efficiency of Markov chain algorithms for sampling.

Suppose we are given a set of input probabilities $\mathcal{P} = \{p_{i,j}\}$ for all $1 \leq i, j \leq n$ with $p_{i,j} = 1 - p_{j,i}$. A natural Markov chain $\mathcal{M}_{\text{nn}}$ over permutations operates by uniformly choosing a pair of adjacent elements, i and j, and putting i ahead of j with probability $p_{i,j}$ and j ahead of i with probability $p_{j,i}$, independent of their current ordering. We call $\mathcal{M}_{\text{nn}}$ the *nearest-neighbor* transposition chain.

The Markov chain $\mathcal{M}_{\text{nn}}$ was among the first Markov chains studied in terms of its computational efficiency for sampling [1, 10, 9]. Its spectral gap has been studied extensively, both in the uniform and biased settings [3, 4, 8, 10, 32].

A central question is under what conditions the spectral gap of $\mathcal{M}_{\text{nn}}$ is an inverse polynomial in n, which implies a polynomial time bound on the *mixing time*, or the time until the chain will be "close" to its stationary distribution. We say $\mathcal{P}$ is *positively biased* if $p_{i,j} \geq 1/2$ for all $i < j$. It is easy to see that this condition is necessary (see, e.g. [4]); however, it is not sufficient, as demonstrated in [4].

In 2003, Fill [13, 14] introduced the following monotonicity conditions: $p_{i,j+1} \geq p_{i,j}$ for all $1 \leq i < j \leq n-1$ and $p_{i-1,j} \geq p_{i,j}$ for all $2 \leq i < j \leq n$. He conjectured that if $\mathcal{P}$ is positively biased and monotone, then the spectral gap of $\mathcal{M}_{\text{nn}}$ is an inverse polynomial in n, and moreover that the smallest spectral gap is attained at the uniform distribution.

Despite significant work on this subject, Fill's conjecture remains mostly open after more than 15 years. In the uniform case, there is a clever path coupling argument that achieves tight bounds on the mixing time [32]. Various papers [3, 4] have identified different classes of $\mathcal{P}$ for which $\mathcal{M}_{\text{nn}}$ may cleverly be viewed as the direct product of simpler independent Markov chains, and thus may be analyzed easily in terms of those chains. In [3], the authors proved a bound of $O(n^2)$ on the mixing time of $\mathcal{M}_{\text{nn}}$ in the case that $p_{i,j} = p$ for all $i < j$, and in [4] the authors considered the case that $p_{i,j}$ depends only on the smaller of i and j.

Among the positively biased, monotone distributions that have proven to be challenging to analyze are those arising in the context of self-organizing lists, where each element i has a frequency w_i of being requested, and then moved ahead one in the list; in this case, $p_{i,j} = w_i/(w_i + w_j)$. The Markov chain $\mathcal{M}_{\text{nn}}$ was termed a "gladiator chain" in this case by Haddadan and Winkler [16].



Figure 1 An example of a 4-class permutation and corresponding 4-particle process, with $n = 7$.

It is instructive to consider *k-classes* [25], where $[n]$ is partitioned into k classes and particles from class i and class j interact with a fixed probability $\overline{p}_{i,j}$[1]. When $k = n$, this captures the general setting. In this context, $\mathcal{M}_{\text{nn}}$ can be seen as having dual duties: *whisking*,

[1] The bar in the notation indicates that we have re-indexed the probabilities by class.

which uniformly mixes particles of the same type[2], and *sifting*, which mixes particles of different types in together [16]. As the mixing properties of the uniform chain are well-understood, it is sufficient to analyze the sifting operation in isolation [16, 25]. By discarding moves between particles in the same class, we are left with a linear k-particle process that maintains elements within each particle class in fixed relative orders (and therefore we may drop their individual labels and re-index, identifying all elements from class i with the label i, as is done in Figure 1).

In this paper, we use the version of the k-particle process introduced in [25], called $\mathcal{M}_{\text{pp}}$, which is also allowed to make certain non-adjacent transpositions – it may swap i and j if all elements between them are smaller than both i and j. This simplifies our analysis, and as in [25], we compensate by using comparison techniques [9, 28] when evaluating the spectral gap of $\mathcal{M}_{\text{nn}}$.

The *bias* towards having a particle of type i ahead of a particle of type j is $\overline{q}_{i,j} = \overline{p}_{i,j}/\overline{p}_{j,i}$. We say that the bias is *bounded* if there exists a constant $q > 1$ such that $\overline{q}_{i,j} \geq q$ for all $i < j$. After a series of papers [16, 25], it was shown that the spectral gap of $\mathcal{M}_{\text{pp}}$ is at least $\Omega(n^{-2(k-1)})$ for positively biased, monotone, and bounded distributions. These results apply to the gladiator chain (self-organizing lists) with k distinct frequencies. In fact, the result in [25] requires only *weak monotonicity*, and not full monotonicity. Weak monotonicity in the setting of k-classes is defined as follows.

▶ **Definition 1** ([4]). *If $\mathcal{P}$ forms a k-class, then $\mathcal{P}$ is weakly monotonic if properties 1 and either 2 or 3 are satisfied.*

1. $\overline{p}_{i,j} \geq 1/2$ *for all* $1 \leq i < j \leq k$, *and*
2. $\overline{p}_{i,j+1} \geq \overline{p}_{i,j}$ *for all* $1 \leq i < j \leq k-1$ *or*
3. $\overline{p}_{i-1,j} \geq \overline{p}_{i,j}$ *for all* $2 \leq i < j \leq k$.

The aforementioned results are based on a natural decomposition of $\mathcal{M}_{\text{pp}}$ into simpler chains, but not as a direct product. To get a bound on the overall spectral gap, the authors of [25] used the decomposition theorem of [23], which bounds the spectral gap of a Markov chain in terms of the spectral gaps of the simpler Markov chains. Unfortunately, one incurs an extra factor of n^{-2} each time it is applied in this setting, and in [25] it is applied iteratively $k-2$ times. Thus, this produced a bound on the spectral gap of $\Omega(n^{-2(k-1)})$, which is an inverse polynomial only for constant k.

To make this iterated decomposition scheme work for larger k requires a stronger decomposition theorem, and that is the main focus of the present paper. We introduce a new decomposition theorem that allows us to achieve nearly optimal bounds of $\Omega(n^{-2})$ on the spectral gap of $\mathcal{M}_{\text{pp}}$ for bounded k-classes, as long as the number of particles of each type is at least $C_q \log n$ (where C_q is a constant depending on the minimum bias q; i.e. not depending on n). We believe this new decomposition theorem is of independent interest and will have other applications.

1.1 The decomposition method

The decomposition method was first introduced by Madras and Randall [21], and has been subsequently used and modified to produce the first polynomial time bounds on the spectral gaps of many interesting Markov chains [6, 7, 11, 12, 15, 16, 18, 22, 23, 24, 26, 27]. Suppose $\mathcal{M}$ is a finite, ergodic Markov chain that is reversible and has stationary distribution π.

[2] We use the terms "type" and "class" interchangeably.

Let $\Omega = \cup_{i=1}^{r} \Omega_i$ be a partition of the state space of $\mathcal{M}$ and γ_i be the spectral gap of $\mathcal{M}$ restricted to Ω_i. The disjoint decomposition theorem of [23] states that the spectral gap γ of $\mathcal{M}$ satisfies $\gamma \geq \frac{1}{2}\gamma_{\min}\bar{\gamma}$, where $\gamma_{\min} = \min_i \gamma_i$ and $\bar{\gamma}$ is the spectral gap of a certain *projection* chain over states $[r]$.

There has been significant effort towards improving the decomposition technique by providing stronger bounds in special cases [7, 12, 15, 18, 22, 23, 26, 27]. While γ may indeed be on the order of $\gamma_{\min}\bar{\gamma}$ – one example is the random walk on the path, decomposed into two smaller paths – there are instances in which it may instead satisfy the much larger bound $\gamma \geq c\min\{\gamma_{\min}, \bar{\gamma}\}$, for some constant c. The simplest such example is the direct product of two independent Markov chains [4, 12]; in this case, $c = 1$.

Tight bounds are especially important when applying the decomposition method iteratively, as was done in [25]. At each level of the induction, $\bar{\gamma} = \Theta(n^{-2})$, so the original bound of [23] yields $\gamma = \Omega(n^{-2(n-1)})$ for the final iteration. Even a bound of the form $\gamma \geq c\min\{\gamma_{\min}, \bar{\gamma}\}$, such as the one in [26], would introduce a factor of c for each application, and would thus yield a bound that is an inverse exponential in n if $c < 1$ is constant. The bounds in [18] are iterable in some cases, but $\mathcal{M}_{\text{pp}}$ does not satisfy those conditions.

1.2 Our results

In this paper, we develop a new decomposition framework that yields iterable bounds for a new class of Markov chains. Among our results, we present a complementary decomposition theorem, which achieves a tight bound on γ without appealing to a bound on the gap $\bar{\gamma}$ of the projection chain, but rather the minimum gap $\tilde{\gamma}_{\min}$ of certain *complementary restrictions* $\tilde{P}_1, \tilde{P}_2, \ldots, \tilde{P}_{\tilde{r}}$. We first consider the simple setting where the state space Ω can be seen as a product space, i.e. $\Omega = \Omega_1 \times \Omega_2$. In other words, for each $a \in \Omega_1$ and each $b \in \Omega_2$, there is a unique $\sigma = (a, b) \in \Omega$. This setting is similar to the direct product of independent Markov chains, but the transition probabilities are not necessarily independent. We define a restriction chain P_a for each $a \in \Omega_1$ that fixes a and operates only on the second coordinate. Similarly, we define a complementary restriction chain $\tilde{P}_b$, which fixes b and operates only on the first coordinate. Recall π is the stationary distribution of $\mathcal{M}$. We write $\pi(a) = \sum_{b\in\Omega_2} \pi(a, b)$ and $\pi(b) = \sum_{a\in\Omega_1} \pi(a, b)$. Define

$$r(a, b) = \frac{\pi(a, b)}{\pi(a)\pi(b)}.$$

The function $r(a, b)$ allows us to capture the degree of dependence between a and b. Let

$$\epsilon = \sum_{(a,b)\in\Omega} \pi(a, b) \left(\sqrt{r(a, b)} - 1/\sqrt{r(a, b)}\right)^2. \tag{1}$$

We say a decomposition satisfying the equality above is ϵ-orthogonal.

▶ **Theorem 2.** *For any ϵ-orthogonal decomposition of Markov chain $\mathcal{M}$ on product space Ω,*

$$\gamma(\mathcal{M}) \geq \min\{\gamma_{\min}, \tilde{\gamma}_{\min}\} \left(1 - \sqrt{\epsilon}\right)^2.$$

This bound can be iterated t times with only a constant overhead, as long as $\sqrt{\epsilon} \leq 1/t$. We note that parts of the proof of this theorem are similar to a "multi-decomposition" result of Destainville [7], which we discuss in Section 5. There we also present several generalizations of Theorem 2, which apply beyond the product space setting.

Favorably, analysis of ϵ-orthogonality requires only a comparison between two stationary distributions and not an analysis of the dynamics of any Markov chain. When $\mathcal{M}$ is a direct product of independent Markov chains, we have that $r(a,b) = 1$ for all pairs $a \in \Omega_1$ and $b \in \Omega_2$ and the decomposition is 0-orthogonal, leading to the bound $\gamma \geq \min\{\gamma_{\min}, \tilde{\gamma}_{\min}\}$, as expected. However, we do not require a strong pointwise bound on $r(a,b)$. The notion of ϵ-orthogonality captures the *average* value of $r(a,b)$, and allows us to achieve tight bounds on γ even when the constituent Markov chains are only *nearly* independent. Indeed, it is possible to prove ϵ is very small even if $r(a,b)$ is far from 1 for pathological pairs a and b, as long as it is close to 1 on average. Importantly, this holds even though the elements in this "bad" space are visited polynomially often.

Armed with our new decomposition theorem, we use the iterated decomposition in [25] to achieve nearly optimal bounds on the spectral gap of the k-particle process $\mathcal{M}_{\text{pp}}$. We prove that this decomposition is $1/n^2$-orthogonal at each level of the decomposition. Thus, we may apply Theorem 2 iteratively k times to get a bound of $\Omega(n^{-2})$ on the spectral gap $\gamma(\mathcal{M}_{\text{pp}})$. This bound is optimal up to constants. More formally, let $N^* = \max\left\{\frac{\log(6n^2)+\log((1+q)/(q-1))}{\log((1+q)/2)}, \frac{\log^2(14)}{\log^2(2q/(1+q))}\right\} =: C_q \log n$, and let c_i denote the number of particles of type i, for $1 \leq i \leq k$. We prove the following.

▶ **Theorem 3.** *If the probabilities $\mathcal{P}$ are weakly monotonic and bounded and $c_i \geq 2N^*$ for all $1 \leq i \leq k$, then the spectral gap γ of the chain $\mathcal{M}_{pp}$ satisfies $\gamma = \Omega(n^{-2})$.*

4131124131342

Figure 2 At level 2 of the decomposition, all particles of type 1 are in fixed positions, and the underlined particle 2 may swap with the 3 to its left or the 4 immediately to its right.

The iterated decomposition works as follows. At the ith level of the decomposition, all particles of type less than i are in fixed positions, and particles of larger type are allowed to swap across these (this is the reason for allowing these non-adjacent transpositions); see Figure 2. This decomposition is designed to exploit the hypothesis that the movement of the particles of type i is nearly independent of the relative order of the particles of type bigger than i. The tool of ϵ-orthogonality allows us to make this intuition precise. We define the complementary restriction chains to contain the moves involving only particles of type bigger than i, and we define the restriction chains to contain moves between particles of type i and particles of larger type. We prove that this decomposition is $1/n^2$-orthogonal if the number of particles of type $i+1$ is large enough. Indeed, the highest probability configuration is the one in which particles are sorted by class, with all particles of smaller type appearing before particles of larger type. Thus, having many particles of type $i+1$ ensures that a typical configuration will not have a particle of type i after a particle of type at least $i+2$, as this requires many transpositions from the highest probability configuration, and each costs a factor of at least the minimum bias q; see Figure 3. Note that such "bad" configurations are polynomially suppressed, but not exponentially suppressed.

1223123111213341314344 4

Figure 3 A "typical" configuration at level 2 has no 2's appearing after any $j \geq 4$.

We use Theorem 3 and comparison techniques to prove the following result for $\mathcal{M}_{\text{nn}}$.

▶ **Theorem 4.** *If the probabilities $\mathcal{P}$ are weakly monotonic and form a bounded k-class with at least $2N^*$ particles in each class, then $\gamma(\mathcal{M}_{nn}) = \Omega(n^{-7})$.*

This is the first inverse-polynomial bound on the spectral gap of $\mathcal{M}_{\text{nn}}$ for monotone bounded k-classes for k as large as $\Theta(n/\log n)$. This is a significant improvement over the previous result which was inverse polynomial only for constant k [23]. Theorem 4 also leads to a bound of $O(n^9)$ on the mixing time of $\mathcal{M}_{\text{nn}}$ under the same conditions.

1.3 Outline

The layout of the paper is as follows. In Section 2, we begin with some notation and terminology. In Section 3, we give details on Theorem 2 and illustrate its use by applying it to the one-dimensional Ising model. In Section 4, we apply Theorem 2 to the biased permutation problem. In Section 5, we generalize the notion of ϵ-orthogonality to non-product spaces and present a more general complementary decomposition theorem that applies to all ϵ-orthogonal decompositions. We also present a classical decomposition theorem that generalizes some results of [18] and show how our decomposition theorems relate to previous results. Finally, we give a brief summary of the proof techniques for the decomposition theorems. Appendix A provides even more detail concerning the proofs of the decomposition theorems, whereas the complete proofs appear in the full version of the paper.

2 Preliminaries

We assume $\mathcal{M}$ is an ergodic Markov chain over a finite state space Ω with transition matrix P. We also assume $\mathcal{M}$ is reversible and has stationary distribution π; that is, it satisfies the *detailed balance* condition: for all $\sigma, \tau \in \Omega$, $\pi(\sigma)P(\sigma,\tau) = \pi(\tau)P(\tau,\sigma)$. Notationally, we write $\pi(S) = \sum_{\sigma\in S}\pi(\sigma)$ for any set $S \subseteq \Omega$.

Let $\Omega = \cup_{i=1}^{r}\Omega_i$ be a partition of the state space into r pieces. For each $i \in [r]$, define $P_i = P(\Omega_i)$ as the restriction of P to Ω_i which rejects moves that leave Ω_i. Formally, P_i is defined as follows: if $\sigma \neq \tau$ and $\sigma, \tau \in \Omega_i$ then $P_i(\sigma,\tau) = P(\sigma,\tau)$; if $\sigma \in \Omega_i$ then $P_i(\sigma,\sigma) = 1 - \sum_{\tau\in\Omega_i,\tau\neq\sigma} P_i(\sigma,\tau)$. The Markov chain $\mathcal{M}_i$ with transition matrix P_i is called a *restriction* Markov chain, and its state space is Ω_i. Let π_i be the normalized restriction of π to Ω_i; i.e. $\pi_i(S) = \pi(S \cap \Omega_i)/\pi(\Omega_i)$ for any $S \subseteq \Omega$. The chain $\mathcal{M}_i$ is ergodic, reversible, and has stationary distribution π_i.

We will be interested in decomposing P into the part that performs restriction moves and the part that performs all other moves. Define $\tilde{P}$ to be the transition matrix of the Markov chain defined by rejecting moves from σ to τ if σ and τ are within the same Ω_i. The matrix $\tilde{P}$ defines a *complementary* partition $\Omega = \cup_{j=1}^{\tilde{r}}\tilde{\Omega}_j$, where each $\tilde{\Omega}_j$ is a maximal subset of Ω that is connected by $\tilde{P}$. For each $j \in [\tilde{r}]$, define the *complementary restriction* $\tilde{P}_j = P(\tilde{\Omega}_j)$ as the restriction of P to $\tilde{\Omega}_j$ which rejects moves that leave $\tilde{\Omega}_j$. The complementary restriction $\tilde{P}_j$ is also ergodic, reversible, and its stationary distribution is the normalized restriction of π to $\tilde{\Omega}_j$, which we call $\tilde{\pi}_j$. Notice that the complementary restrictions are defined by the decomposition $P_1, P_2, \ldots, P_r$.

The efficiency of a Markov chain $\mathcal{M}$ is a function of its spectral gap, denoted $\gamma(\mathcal{M})$, which is defined as the difference of 1 and the second largest eigenvalue of its transition matrix. Letting $\gamma_i = \gamma(\Omega_i)$ and $\tilde{\gamma}_j = \gamma(\tilde{\Omega}_j)$, the complementary decomposition theorem, Theorem 2, is proven by analyzing the spectral gaps $\gamma_{\min} = \min_i \gamma_i$ and $\tilde{\gamma}_{\min} = \min_j \tilde{\gamma}_j$. Note that if some restriction or complementary restriction has a single element, its spectral gap is taken to be 1.

3 Complementary decomposition theorem

In this section, we show how to apply our new complementary decomposition theorem by considering a few simple examples. Recall $r(a,b) = \pi(a,b)/(\pi(a)\pi(b))$ and

$$\epsilon = \sum_{(a,b)\in\Omega} \pi(a,b)\left(\sqrt{r(a,b)} - 1/\sqrt{r(a,b)}\right)^2.$$

Theorem 2 states that the spectral gap γ of $\mathcal{M}$ satisfies $\gamma \geq \min\{\gamma_{\min}, \tilde{\gamma}_{\min}\}(1-\sqrt{\epsilon})^2$. A simple application of Theorem 2 is to a Markov chain $\mathcal{M}$ that is the direct product of two Markov chains $\mathcal{M}_1$ and $\mathcal{M}_2$. It is easy to see that $r(a,b) = 1$ for all a, b, and so this proves $\gamma(\mathcal{M}) = \min\{\gamma(\mathcal{M}_1), \gamma(\mathcal{M}_2)\}$. By iterating on $\tilde{\gamma}_{\min}$, we can immediately prove the following well-known result.

▶ **Corollary 5.** *If $\mathcal{M}$ is the direct product of Markov chains $\{\mathcal{M}_i\}$, then $\gamma(\mathcal{M}) = \min_i \gamma(\mathcal{M}_i)$.*

3.1 One-dimensional Ising model

As a second introductory example prior to our main application, we consider the one-dimensional Ising model. Here each configuration $\sigma \in \Omega$ is an assignment of a "spin" (either +1 or -1) to each of n vertices connected to form a line; see Figure 4. Let $\lambda = e^{-\beta}$, where $\beta > 0$ represents inverse temperature. We are interested in sampling from the Gibbs distribution given by $\pi(\sigma) = e^{-\beta H(\sigma)}/Z$, where the Hamiltonian $H(\sigma)$ is the number of edges whose endpoints have different spins and Z is the normalizing constant $\sum_{\sigma\in\Omega} e^{-\beta H(\sigma)}$, also called the *partition function*. (See [20] for background on the ferromagnetic Ising model.)

Consider the Glauber dynamics Markov chain $\mathcal{M}_{\text{gd}}$.

Glauber Dynamics $\mathcal{M}_{\text{gd}}$

`Starting at any configuration` σ^0`, iterate the following:`

- At time t, choose a vertex $1 \leq i \leq n$ uniformly at random.
- Set the spin of vertex i to $+1$ with probability $p = (\pi(\sigma^{t,i\leftarrow +}))/(\pi(\sigma^{t,i\leftarrow +}) + \pi(\sigma^{t,i\leftarrow -}))$ where $\sigma^{t,i\leftarrow +}$ is identical to σ^t with the spin of vertex i set to $+1$ (or -1 for $\sigma^{t,i\leftarrow -}$).
- Otherwise, set the spin of vertex i to -1 with probability $1-p$.

For simplicity, we will assume that n is a power of 2. To apply our theorem, we decompose the state space by breaking configurations in half along the middle edge; again, see Figure 4. Transitions that fix the signs on the left are part of the restriction chains, and transitions that fix signs on the right are part of the complementary restriction chains. Thus, our restrictions and complementary restrictions are both $1 \times n/2$ Ising models for which we can readily apply induction. Let a be the assignment of signs to the left $n/2$ vertices and b be the signs of the right $n/2$ vertices. It is straightforward to see $\sigma = (a,b)$ gives a unique configuration and that the state space is a product space. However, $\mathcal{M}_{\text{gd}}$ is not a direct product of independent Markov chains on a and b because the probability of changing a sign of either of the middle two vertices ($n/2$ or $n/2+1$) depends on the sign of the other middle vertex. In order to apply Theorem 2, we first analyze $r(a,b) = \pi(a,b)/(\pi(a)\pi(b))$ and subsequently ϵ. The techniques used here are similar to, but simpler than, those used Section 4.

Define $\lambda = e^{-\beta}$. Let $w(a) = \lambda^{H(a)}$, where $H(a)$ is the number of edges in a (the left half) with disagreeing signs. Analogously, define $w(b) = \lambda^{H(b)}$. Let $c_{ab} = \lambda$ if the middle signs disagree and $c_{ab} = 1$ otherwise. Thus $\pi(a,b) = w(a)w(b)c_{ab}/Z$. Let $\Omega^* \subset \Omega$ be the configurations where the middle two vertices agree. Define $Z_A = \sum_a w(a)$ and $Z_B = \sum_b w(b)$.

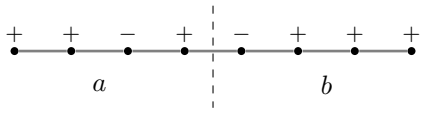


Figure 4 An example configuration of the one-dimensional Ising model.

For any fixed a, we have $\sum_{b:(a,b)\in\Omega^*} w(b) = \sum_{b:(a,b)\notin\Omega^*} w(b) = \frac{1}{2}Z_B$, since we can swap spins on all vertices in b to obtain a unique configuration $b' \in \Omega \setminus \Omega^*$ with $w(b') = w(b)$. Thus, we have

$$Z = \sum_{(a,b)\in\Omega^*} w(a)w(b) + \lambda \sum_{(a,b)\notin\Omega^*} w(a)w(b) = (1+\lambda) \sum_{(a,b)\in\Omega^*} w(a)w(b) = \frac{(1+\lambda)Z_A Z_B}{2}.$$

We consider two different cases for $r(a,b)$ depending on whether the sign of the middle two vertices agree. First consider the case where $(a,b) \in \Omega^*$. Here we have

$$\pi(a) = \sum_{b'} \pi(a,b') = \sum_{b':(a,b')\in\Omega^*} \frac{w(a)w(b')}{Z} + \lambda \sum_{b':(a,b')\notin\Omega^*} \frac{w(a)w(b')}{Z} = \frac{w(a)(1+\lambda)Z_B}{2Z},$$

and similarly $\pi(b) = w(b)(1+\lambda)Z_A/(2Z)$. Therefore

$$r(a,b) = \frac{\pi(a,b)}{\pi(a)\pi(b)} = \frac{4Zw(a)w(b)c_{ab}}{w(a)w(b)(1+\lambda)^2 Z_A Z_B} = \frac{2}{1+\lambda}.$$

The next case is almost identical except that $c_{ab} = \lambda$, so we have for $(a,b) \notin \Omega^*$ that $r(a,b) = 2\lambda/(1+\lambda)$.

Next we use our analysis of $r(a,b)$ to bound ϵ. First notice that since $\sum_{(a,b)\in\Omega^*} \pi(a,b) + \sum_{(a,b)\notin\Omega^*} \pi(a,b) = 1$ and $\sum_{(a,b)\notin\Omega^*} \pi(a,b) = \lambda \sum_{(a,b)\in\Omega^*} \pi(a,b)$, we have that $\sum_{(a,b)\in\Omega^*} \pi(a,b) = 1/(1+\lambda)$ and $\sum_{(a,b)\notin\Omega^*} \pi(a,b) = \lambda/(1+\lambda)$. This yields

$$\begin{aligned}\epsilon &\leq \sum_{(a,b)\in\Omega^*} \pi(a,b)(\sqrt{r(a,b)} - 1/\sqrt{r(a,b)})^2 + \sum_{(a,b)\notin\Omega^*} \pi(a,b)(\sqrt{r(a,b)} - 1/\sqrt{r(a,b)})^2 \\ &= \frac{1}{1+\lambda}\left(\sqrt{\frac{2}{1+\lambda}} - \sqrt{\frac{1+\lambda}{2}}\right)^2 + \frac{\lambda}{1+\lambda}\left(\sqrt{\frac{2\lambda}{1+\lambda}} - \sqrt{\frac{1+\lambda}{2\lambda}}\right)^2 = \left(\frac{1-\lambda}{1+\lambda}\right)^2\end{aligned}$$

Applying the complementary decomposition theorem (Theorem 2) gives the following recurrence: $\gamma_n \geq \gamma_{n/2}\left(\frac{2\lambda}{1+\lambda}\right)^2$. Since the base case has gap $\Omega(n^{-1})$, this solves to $\gamma_n = \Omega(n^{-c})$ for $c = 1 + 2\log_2\left(\frac{1+\lambda}{2\lambda}\right)$. Note that while this does not give a tight bound, the constant c is strictly better than the constant given by [18] and, unlike earlier decomposition approaches, we have not incurred an extra factor of n with each application of the decomposition theorem.

3.2 Ising model on bounded degree trees

As in [18], our proof for the one-dimensional Ising model can be easily generalized to trees with constant maximum degree r. A straightforward induction shows that such a tree T on n vertices has an edge whose deletion cuts T into two components, each with size at least $n/(r+1)$. We let a represent the spins on one component and b the spins on the other. At each level of the induction, we compute $r(a,b)$ and ϵ using arguments similar to those in Section 3.1 to get $\gamma_n = \Omega(n^{-c})$ for $c = 1 + 2\log_{(r+1)/r}\left(\frac{1+\lambda}{2\lambda}\right)$.

4 Application to permutations

In this section, we apply the complementary decomposition theorem, Theorem 2, to the problem of sampling biased permutations. We give an overview of the proof of Theorem 4 which bounds the spectral gap of the following nearest-neighbor Markov chain over S_n, the permutations of $[n]$. The complete details are provided in the full version of the paper.

The Nearest Neighbor Markov chain $\mathcal{M}_{\text{nn}}$

`Starting at any permutation` σ^0`, iterate the following:`

- At time $t \geq 0$, choose a position $1 < i \leq n$ uniformly at random in permutation σ^t.
- With probability $p_{\sigma^t(i),\sigma^t(i-1)}/2$, exchange the elements $\sigma^t(i)$ and $\sigma^t(i-1)$ to obtain σ^{t+1}.
- Otherwise, do nothing so that $\sigma^{t+1} = \sigma^t$.

The chain $\mathcal{M}_{\text{nn}}$ connects the state space S_n and has the following stationary distribution (see e.g., [4]):

$$\pi_{\text{nn}}(\sigma) = \prod_{i<j:\sigma(i)>\sigma(j)} \frac{p_{\sigma(i),\sigma(j)}}{p_{\sigma(j),\sigma(i)}} Z_{\text{nn}}^{-1} = \prod_{i<j:\sigma(i)>\sigma(j)} q_{\sigma(i),\sigma(j)} Z_{\text{nn}}^{-1}$$

where Z_{nn} is a normalizing constant and $q_{\sigma(i),\sigma(j)} = \frac{p_{\sigma(i),\sigma(j)}}{p_{\sigma(j),\sigma(i)}}$.

We consider the special case of k-classes where $[n]$ is partitioned into k classes $C_1, C_2, \ldots, C_k$, and assume elements in class C_i interact with elements in class C_j with the same probability. That is, if $i_1, i_2 \in C_i$ and $j_1, j_2 \in C_j$ then $p_{i_1,j_1} = p_{i_2,j_2}$. In this case we define $\overline{p}_{i,j}$ to be this shared probability for classes C_i and C_j (the bar indicates that we have reindexed the set of probabilities by the classes) and we say that $\mathcal{P}$ forms a *k-class*. Note that $\overline{p}_{i,i}$ is assumed to be $1/2$, so that $\mathcal{M}_{\text{nn}}$ swaps elements within the same class with probability $1/2$. When $k = n$, the k-class assumption does not lose any generality, but this structure allows us to simplify the problem by considering $k < n$, as was done in [25, 16].

Define $\overline{q}_{i,j} = \overline{p}_{i,j}/\overline{p}_{j,i}$ to be the *bias* towards having a particle of type i ahead of a particle of type j. We say that $\mathcal{P}$ is *bounded* if there exists a constant $q > 1$ such that $\overline{q}_{i,j} \geq q$ for all $1 \leq i < j \leq k$. The constant q is called the *minimum bias*. We prove the following.

▶ **Theorem 4.** *If the probabilities $\mathcal{P}$ are weakly monotonic and form a bounded k-class with at least $2N^*$ particles in each class, then $\gamma(\mathcal{M}_{nn}) = \Omega(n^{-7})$.*

The chain $\mathcal{M}_{\text{nn}}$ samples over S_n using these probabilities, and in particular the order of elements within each class approaches the uniform distribution. The spectral gap of this uniform sampling is well-understood and may be analyzed separately. The complete analysis can be found in the full version of the paper. In order to isolate the biased moves, we define a new Markov chain $\mathcal{M}_{\text{pp}}$ that eliminates swaps within each class. As $\mathcal{M}_{\text{pp}}$ maintains a fixed order on particles within each class, it makes sense to relabel each element of $[n]$ by the index of the class it is in. That is, we let $c_i = |C_i|$ and we consider a linear array of length n with c_i particles labeled i for each $1 \leq i \leq k$. We call this a k-particle system for the given set $\{c_i\}$, and the Markov chain $\mathcal{M}_{\text{pp}}$ is called a k-particle process. We view the new state space as the set Ω of k-particle systems for $\{c_i\}$.

The Markov chain $\mathcal{M}_{\text{pp}}$ over k-particle systems also allows certain non-adjacent transpositions. In particular, we let a particle of type i and a particle of type j swap across particles of type less than i and j. More formally, the chain $\mathcal{M}_{\text{pp}}$ is defined as follows.

The particle process Markov chain $\mathcal{M}_{\text{pp}}$

`Starting at any` k`-particle system` σ^0`, iterate the following:`

- At time t, choose a position $1 \le i \le n$ and direction $d \in \{L, R\}$ uniformly at random.
- If $d = L$, find the largest j less than i with $\sigma^t(j) \ge \sigma^t(i)$ (if one exists). If $\sigma^t(j) > \sigma^t(i)$, then with probability $1/2$, exchange $\sigma^t(i)$ and $\sigma^t(j)$ to obtain σ^{t+1}.
- If $d = R$, find the smallest j with $j > i$ and $\sigma^t(j) \ge \sigma^t(i)$ (if one exists). If $\sigma^t(j) > \sigma^t(i)$, then exchange $\sigma^t(i)$ and $\sigma^t(j)$ to obtain σ^{t+1} with probability

$$\frac{1}{2}\,\overline{q}_{\sigma_t(j),\sigma_t(i)} \prod_{i<l<j} \overline{q}_{\sigma_t(j),\sigma_t(l)}\overline{q}_{\sigma_t(l),\sigma_t(i)}.$$

- With all remaining probability, $\sigma^{t+1} = \sigma^t$.

The chain $\mathcal{M}_{\text{pp}}$ connects the space Ω and has the stationary distribution (see e.g., [4])

$$\pi(\sigma) = \prod_{i<j:\sigma(i)>\sigma(j)} \frac{\overline{p}_{\sigma(i),\sigma(j)}}{\overline{p}_{\sigma(j),\sigma(i)}} Z^{-1} = \prod_{i<j:\sigma(i)>\sigma(j)} \overline{q}_{\sigma(i),\sigma(j)} Z^{-1}$$

where Z is a normalizing constant and $\overline{q}_{\sigma(i),\sigma(j)} = \frac{\overline{p}_{\sigma(i),\sigma(j)}}{\overline{p}_{\sigma(j),\sigma(i)}}$.

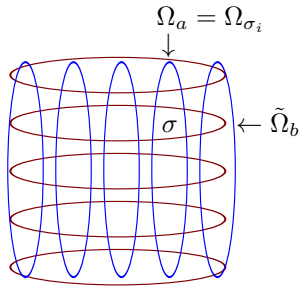
Recall the definition of weakly monotonic from Section 1. We will assume that property (2) holds. If instead property (3) holds, then as described in [25] we would modify $\mathcal{M}_{\text{pp}}$ to allow swaps between elements of different particle types across elements whose particle types are larger (instead of smaller) and modify the induction so that at each step σ_i restricts the location of particles larger than i (instead of smaller).

We prove the following bound on the spectral gap of $\mathcal{M}_{\text{pp}}$. Then, using comparison techniques [9, 28], we prove the bound on the spectral gap of $\mathcal{M}_{\text{nn}}$ given in Theorem 4. The details of this comparison argument can be found in the full version of the paper.

▶ **Theorem 3.** *If the probabilities $\mathcal{P}$ are weakly monotonic and bounded and $c_i \ge 2N^*$ for all $1 \le i \le k$, then the spectral gap γ of the chain $\mathcal{M}_{pp}$ satisfies $\gamma = \Omega(n^{-2})$.*

The proof of Theorem 3 uses the same inductive technique as [25], where at each level of the induction we fix the locations of particles in one less particle class. For $i \ge 0$, let σ_i represent a fixed location of the particles of type $1, 2, \ldots, i$ (σ_0 represents no restriction); for example, in Figure 5, we set $\sigma_2 = 12_1_2_1_\ _$, where "_" represents locations that can be filled with particles of type 3 or higher. We consider the chain $\mathcal{M}_{\sigma_i}$ whose state space Ω_{σ_i} is the set of all k-particle systems σ where the elements with type in $[i]$ are consistent with σ_i. The moves of $\mathcal{M}_{\sigma_i}$ are those moves from $\mathcal{M}_{\text{pp}}$ that do not involve an element of type at most i. We prove by induction that $\mathcal{M}_{\sigma_i}$ has spectral gap $\Omega(n^{-2}(1-1/n)^{2(k-2-i)})$ for all choices of σ_i. To be clear, we assume that the spectral gap of $\mathcal{M}_{\sigma_i}$ are bounded for all σ_i by induction, and then prove our bound on the spectral gap of $\mathcal{M}_{\sigma_{i-1}}$. To start, we show that $\Omega_{\sigma_{i-1}}$ is a product space, which is required to apply Theorem 2.

Let A consist of all 2-particle systems with c_i particles of type i and $\sum_{j=i+1}^{k} c_j$ particles of type "_"; A is in bijection with staircase walks by mapping each i to a step right and each "_" to a step down, as in Figure 6. Let B consist of all $k-i$ particle systems with c_j particles of type j for all $i+1 \le j \le k$. Our goal is to show that the set of permutations σ consistent with σ_{i-1} on particles of type at most $i-1$ is in bijection with $A \times B$. To this end, we can write $\sigma = (a, b)$, where $a \in A$ is the 2-particle system obtained from σ_i by removing particles of type less than i (see Figure 5), as those particles are in a fixed position for all of $\Omega_{\sigma_{i-1}}$.



$$\begin{aligned} \sigma_2 &= 12_1_2_1_\ _ \\ \sigma_3 &= 1231_2_13_ \\ a &= 3_\ _3_ \\ b &= 454 \\ \sigma &= 123142513 4 \end{aligned}$$

Figure 5 The state space $\Omega_{\sigma_{i-1}}$ decomposed, with $i = 3$.

Next, define $b \in B$ to be the restriction of σ to elements of type bigger than i. For the other direction, given any (a, b) pair, it is clear that there is a unique $\sigma \in \Omega_{\sigma_{i-1}}$ corresponding to that pair.

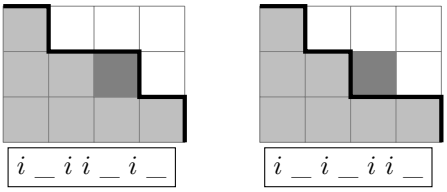


Figure 6 An exclusion process on staircase walks operates by adding or removing a square.

We next describe the decomposition. Note that the moves of $\mathcal{M}_{\sigma_i}$ fix an $a \in A$ and perform (j_1, j_2) transpositions, where $j_1, j_2 > i$; i.e. they operate exclusively on B. Thus, the Markov chain $\mathcal{M}_{\sigma_i}$ is a restriction of $\mathcal{M}_{\sigma_{i-1}}$ with state space Ω_a. On the other hand, the remaining moves of $\mathcal{M}_{\sigma_{i-1}}$ are (i, j) transpositions for $j > i$. These are the complementary restrictions; these moves fix a $b \in B$ and operate on A, so we label the state space of this Markov chain $\tilde{\Omega}_b$. As these moves fix the relative order of all particles of type bigger than i, the complementary restriction chains can be seen as bounded exclusion processes on particles of type i with particles of type bigger than i. Bounded generalized biased exclusion processes operate on staircase walks as in Figure 6, where every square has a different bias but they are all bounded by some q. These processes were analyzed in [25]. We prove the following lemma in the full version of the paper.

▶ **Lemma 6.** *The complementary restrictions at each level of the induction are bounded generalized biased exclusion processes with spectral gap* $\Omega(n^{-2})$.

The chain $\mathcal{M}_{\sigma_{i-1}}$ is not the direct product of the chains on A and B because, e.g., for $(a, b) \in A \times B$, $P((a, b), (a', b))$ depends on b. However, we show that the above decomposition is $1/n^2$-orthogonal by bounding $r(a, b)$. We define "good" a's to be those with fewer than $N^* = C_q \log n$ inversions and "good" b's to be those with fewer than N^* inversions involving particles of type $i + 1$. Thus, as there are at least $2N^*$ particles of type $i + 1$, then (a, b) has no inversions between i and j for $j > i + 1$ when a and b are both good. For such pairs, $r(a, b)$ is very close to 1. For all other pairs we show $r(a, b)\pi(\Omega_a \cap \tilde{\Omega}_b)$ is small. By viewing b as a staircase walk on particles of type $i + 1$ with particles of any higher type, we see that for either a or b, the probability it is bad is smaller than the weighted sum of all biased exclusion processes with more than N^* inversions (equivalently, area N^* under the curve). We prove the following lemma in the full version of the paper.

▶ **Lemma 7.** *For any biased exclusion process with minimum bias constant $q > 1$, the total weight of staircase walks with area larger than N^* satisfies*

$$\sum_{\sigma: N(\sigma) \geq N^*} q^{-N(\sigma)} \leq \frac{1}{6n^2}.$$

This allows us to prove the following.

▶ **Lemma 8.** *If the probabilities $\mathcal{P}$ are weakly monotonic and bounded with $c_i \geq 2N^*$ for all $1 \leq i \leq k$, then at each step of the induction we have a $1/n^2$-orthogonal decomposition.*

We illustrate the main ideas of the proof of Lemma 8 here in the simplified $k = 3$ case (the details are deferred to the full version). In this case, there is no recursion but instead just a single application of the decomposition theorem. The restriction chains of $\mathcal{M} = \mathcal{M}_{\sigma_0}$ are the set $\{\mathcal{M}_{\sigma_1}\}$, which fix all elements in class C_1. The stationary distribution of $\mathcal{M}$ is

$$\pi(\sigma) = Z^{-1} \prod_{\substack{i<j: \\ \sigma(i)>\sigma(j)}} \overline{q}_{\sigma(i),\sigma(j)}, \tag{2}$$

where Z is a normalizing constant.

Let $w(a)$ and $w(b)$ be the parts of this product that depend only on a and only on b, respectively, and let $w(a, b)$ be a correction factor that depends on both a and b. Let $t_{1,3}$ (respectively, $t_{1,2}$ and $t_{2,3}$) denote the number of inversions in σ between a 1 and a 3 (respectively a 1 and a 2 and a 2 and a 3). For example, let $\sigma = 111221312323$, which has stationary probability $Z^{-1}(\overline{q}_{1,2})^4(\overline{q}_{2,3})^3(\overline{q}_{1,3})$. Then $a = 111_\ _1_1_\ _\ _\ _\ b = 2232323$. From b we find that $t_{2,3} = 3$ and $w(b) = (\overline{q}_{2,3})^3$; more generally, define $w(b)$ to be the product $(\overline{q}_{2,3})^{t_{2,3}}$. From a, we can see that there are five inversions involving 1, but the number of those that are inversions with a 3 versus a 2 depends on b as well. Ignoring this for a moment, we define $w(a)$ to be the product $(\overline{q}_{1,2})^{t_{1,2}+t_{1,3}}$. In our example, $w(a) = (\overline{q}_{1,2})^5$. Since we have made the false assumption that there were no inversions between a 1 and a 3 in σ, we need a correction factor $w(a, b) = (\overline{q}_{1,3}/\overline{q}_{1,2})^{t_{1,3}}$. With these definitions, it is clear that $\pi(\sigma) = Z^{-1}w(a)w(b)w(a, b)$.

A key idea in the proof of Lemma 8 is that if a and b are both good, then $t_{1,3} = 0$ – indeed, the total number of inversions is less than $2N^*$ and the number of 2's is at least $2N^*$ – and thus the correction factor $w(a, b) = 1$, implying $\pi(a, b) \approx \pi(a)\pi(b)$. Moreover, the probability that a or b is bad is very small, so these pairs (a, b) do not contribute much to the sum in Equation 1.

To make the above statements precise is somewhat technical. Deferring details to the full version, we now give a bit more intuition. Define $Z_A = \sum_a w(a)$, $Z_B = \sum_b w(b)$, and $\epsilon_1 = 1/(6n^2)$. We show that $\sum_{a \text{ bad}} w(a) \leq \epsilon_1$ and $\sum_{b \text{ bad}} w(b) \leq \epsilon_1/Z_B$. Thus, we find $Z_A \approx \sum_{a \text{ good}} w(a)$ and $Z_B \approx \sum_{b \text{ good}} w(b)$, and moreover $Z = \sum_{a,b} w(a)w(b)w(a, b) \approx Z_A Z_B$. We show that when a and b are both good, $\pi(a) \approx w(a)Z_B/Z$ and $\pi(b) \approx w(b)Z_A/Z$. Thus, when a and b are both good, $r(a, b) = \frac{\pi(a,b)}{\pi(a)\pi(b)} \approx \frac{Z}{Z_A Z_B} \approx 1$. This allows us to show

$$\sum_{\substack{a \text{ good} \\ b \text{ good}}} \pi(a, b) \left(\sqrt{r(a, b)} - \frac{1}{\sqrt{r(a, b)}} \right)^2 \leq 5\epsilon_1^2.$$

We must also consider the case that either a or b is bad. In this case, we show that the weight of these configurations is so small that it overcomes the fact that $w(a, b)$ and $r(a, b)$ may be exponentially small. We use the loose bound

$$\sum_{a \text{ or } b \text{ bad}} \pi(a,b)\left(\sqrt{r(a,b)} - \frac{1}{\sqrt{r(a,b)}}\right)^2 \leq \sum_{a \text{ or } b \text{ bad}} \pi(a)\pi(b) + \sum_{a \text{ or } b \text{ bad}} \frac{\pi(a,b)^2}{\pi(a)\pi(b)}.$$

The summation on the left is bounded by $\Pr(a \text{ bad}) + \Pr(b \text{ bad}) \leq 2\epsilon_1/(1-\epsilon_1)$. The summands in the summation on the right are products of conditional probabilities. We use the law of total probability to bound that summation by $\sum_{a \text{ bad}} w(a) + \sum_{b \text{ bad}} w(b)/(1-\epsilon_1) \leq \epsilon_1 + \epsilon_1/(1-\epsilon_1)$. Putting this all together, we have, for $\epsilon_1 \leq .225$,

$$\sum_{a,b} \pi(a,b)\left(\sqrt{r(a,b)} - \frac{1}{\sqrt{r(a,b)}}\right)^2 \leq 5\epsilon_1^2 + \epsilon_1 + \frac{3\epsilon_1}{1-\epsilon_1} \leq 6\epsilon_1 = 1/n^2.$$

This shows that the decomposition is $1/n^2$-orthogonal.

5 Generalizations and comparison with other theorems

In Sections 5.1 and 5.2, we present several generalizations of Theorem 2 and compare these results with related prior work. Our decomposition theorems fall into two categories: complementary decomposition theorems that rely on the notion of ϵ-orthogonality between the restrictions and complementary restrictions, and more classical decomposition theorems based on the projection Markov chain. In Section 5.3, we summarize the proofs, with more details given in Appendix A.

5.1 Generalized complementary decomposition theorems

Theorem 2 generalizes easily to non-product spaces. Define $r(i,j) = \pi(\Omega_i \cap \tilde{\Omega}_j)/(\pi(\Omega_i)\pi(\tilde{\Omega}_j))$, for any $1 \leq i \leq r, 1 \leq j \leq \tilde{r}$. We say that $\{\Omega_i\}$ and $\{\tilde{\Omega}_j\}$ is an ϵ-orthogonal decomposition of $\mathcal{M}$ if

$$\epsilon = \sum_{(i,j)} \pi(\Omega_i \cap \tilde{\Omega}_j)(\sqrt{r(i,j)} - 1/\sqrt{r(i,j)})^2.$$

▶ **Theorem 9.** *For any ϵ-orthogonal decomposition of $\mathcal{M}$, $\gamma(\mathcal{M}) \geq \min\{\gamma_{\min}, \tilde{\gamma}_{\min}\}\left(1 - \sqrt{\epsilon}\right)^2$.*

We use Theorem 10 to prove Theorem 9, which in turn implies Theorem 2.

▶ **Theorem 10.** $\gamma(\mathcal{M}) \geq \min_{x \perp \sqrt{\pi}, \|x\|=1} \gamma_{\min}\|x^\perp\|^2 + \tilde{\gamma}_{\min}\|\tilde{x}^\perp\|^2$.

Here, $x^\perp$ and $\tilde{x}^\perp$ are orthogonal projections of a vector x onto the complement of the eigenspace of the top eigenvectors of certain matrices (defined in Section A.4) containing the P_i's and $\tilde{P}_j$'s, respectively. This theorem is similar to a special case of the main result in [7]. Destainville [7] introduced a "multi-decomposition" scheme that uses m different partitions of Ω. In Destainville's result, $\|x^\perp\|^2 + \|\tilde{x}^\perp\|^2$ is replaced by a function of the norm of a "multi-projection" operator Π. Bounding these norms is essential, as the Markov chain $\mathcal{M}$ can require exponential time to mix even if all of the restrictions and complementary restrictions are polynomially mixing[3].

[3] Indeed, the introduction of the projection chain in [21] was a key insight to the original decomposition theorem.

Unfortunately, bounding these norms can be challenging. Destainville [7] bounds the norm of the projection Π by the spectral gap of a smaller matrix $\bar{\Pi}$. In some cases, this gap can be analyzed directly, or even computationally for particular problem instances. However, for very complex distributions such as the distribution over biased permutations we consider here, it can be challenging to find the spectral gap of $\bar{\Pi}$. We believe one of our main contributions is the definition of ϵ-orthogonality, a concrete combinatorial quantity that may be easier to analyze. This approach is particularly useful when the chain decomposes into pieces that are nearly independent, as in the setting of Theorem 2.

5.2 Classical decomposition theorems

The disjoint decomposition theorem of [23] states that the spectral gap γ of $\mathcal{M}$ satisfies $\gamma \geq \frac{1}{2}\gamma_{\min}\bar{\gamma}$, where, as we recall from Section 1, $\gamma_{\min} = \min_i \gamma_i$ and $\bar{\gamma}$ is the spectral gap of a projection chain over states $[r]$. Jerrum, Son, Tetali, and Vigoda [18] considered two quantities related to the spectral gap: the Poincaré and log-Sobolev constants. There, the authors defined a parameter $T = \max_i \max_{\sigma\in\Omega_i} \sum_{\tau\in\Omega\setminus\Omega_i} P(\sigma,\tau)$, which can be seen as the maximum probability of escape from one part of the partition in a single step of P, and used it to produce a bound on the order of the minimum gap when T is on the order of $\bar{\gamma}$. They also provided improved bounds when another parameter η is close to zero; this requires a pointwise regularity condition. More recently, Pillai and Smith [26] introduced other conditions in order to directly bound the mixing time by a constant times the maximum of the mixing times of the projection and the restrictions.

The techniques developed for proving the complementary decomposition theorems introduced in this paper can be further applied to prove the following "classical"-style decomposition theorem.

▶ **Theorem 11.** *Let* $\rho = \sqrt{2T/\bar{\gamma}}$. *Then* $\gamma(\mathcal{M}) \geq \min_{p^2+q^2=1} \gamma_{\min} q^2 + \bar{\gamma}\left(q\rho - p\right)^2$.

We state a more general version of this theorem, Theorem 17, in Section A.3. This bound allows us to rederive several known classical decomposition theorems.

▶ **Corollary 12.** *Assume* $\mathcal{M}$ *is lazy. Then* $\gamma \geq \gamma_{\min}\bar{\gamma}/3$.

In fact, one can show that the constant is $1/2$ if $\gamma_{\min}, \bar{\gamma} \leq 1/2$ (which is a common situation) or $\delta_2 \geq 1/2$ (δ_2 is defined in Section A.2). In Corollary 13 we show that Theorem 11 can be seen as a generalization of Theorem 1 of [18], except that it instead bounds the spectral gap.

▶ **Corollary 13.** $\gamma \geq \min\left\{\frac{\bar{\gamma}}{3}, \frac{\gamma_{\min}\bar{\gamma}}{3T+\bar{\gamma}}\right\}$.

In particular, if $T/\bar{\gamma}$ is a constant, then we get within a constant of the minimum gap as well. Theorem 11 produces slightly improved bounds over Corollary 13 when $T \approx \bar{\gamma} \ll \gamma_{\min}$.

5.3 Summary of the proofs of the decomposition theorems

Our proofs are elementary and use only basic facts from linear algebra about eigenvalues and eigenvectors. We have chosen to assume the Markov chains are discrete and finite to keep the proofs as accessible as possible. We utilize the following standard characterization of the second largest eigenvalue λ of a symmetric matrix A with top eigenvector v:

$$\lambda = \max_{x\perp v} \frac{\langle x, xA\rangle}{\|x\|^2} = \max_{x\perp v:\|x\|=1} \langle x, xA\rangle. \tag{3}$$

For a general reversible Markov chain with transition matrix P, we apply Equation 3 to a symmetric matrix $A = A(P)$ that has the same eigenvalues as P.

We apply the Vector Decomposition Method from the expander graph literature (see, e.g. [29, 31]), and decompose the vector x into $x^{\perp} + x^{\parallel}$, where $x^{\parallel}$ is parallel to the top eigenvector of each restriction matrix. The intuition of this method is that if a particular distribution is far from stationarity, then it will either be far from stationarity on some part of the partition or on the projection, and therefore applying P brings us closer to stationarity. The benefit of this approach is that it allows us to quantify the independence of the restriction chains with the projection or complementary restriction chains. Using Equation 3, for any $x \perp v$, we need to bound

$$\langle x, xA\rangle = \langle x^{\perp}, x^{\perp}A\rangle + \langle x^{\parallel}, x^{\parallel}A\rangle + 2\langle x^{\perp}, x^{\parallel}A\rangle. \tag{4}$$

It is easy to bound $\langle x^{\perp}, x^{\perp}A\rangle$ and $\langle x^{\parallel}, x^{\parallel}A\rangle$ using ideas from other decomposition results [18, 23]. The term $\langle x^{\perp}, x^{\parallel}A\rangle$ determines whether the decomposed Markov chain is nearly the direct product of two independent Markov chains $\mathcal{M}_1$ and $\mathcal{M}_2$, in which case $\langle x^{\perp}, x^{\parallel}A\rangle \approx 0$ and $\gamma(\mathcal{M}) \approx \min\{\gamma(\mathcal{M}_1), \gamma(\mathcal{M}_2)\}$, or whether they are far from independent, in which case $\langle x^{\perp}, x^{\parallel}A\rangle$ is large and $\gamma(\mathcal{M}) = \Theta(\gamma_{\min}\bar{\gamma})$. The key to our decomposition proofs lies in our bounds on $\langle x^{\perp}, x^{\parallel}A\rangle$, which are different for our complementary decomposition theorems than they are for our classical decomposition theorems. More details are provided in the appendix.

References

1. D. Aldous. Random walk on finite groups and rapidly mixing Markov chains. *In Seminaire de Probabilites XVII*, pages 243–297, 1983.
2. D. Bayer and P. Diaconis. Trailing the dovetail shuffle to its lair. *The Annals of Applied Probability*, 2:295–313, 1992.
3. I. Benjamini, N. Berger, C. Hoffman, and E. Mossel. Mixing times of the biased card shuffling and the asymmetric exclusion process. *Trans. Amer. Math. Soc*, 357:3013–3029, 2005.
4. P. Bhakta, S. Miracle, D. Randall, and A. P. Streib. Mixing times of Markov chains for self-organizing lists and biased permutations. In *Proceedings of the 24th ACM/SIAM Symposium on Discrete Algorithms*, SODA '13, pages 1–15, 2013.
5. F. Chierichetti, A. Dasgupta, R. Kumar, and S. Lattanzi. On reconstructing a hidden permutation. *Leibniz International Proceedings in Informatics, LIPIcs*, 28:604–617, September 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.604.
6. C. Cooper, M. Dyer, A. Frieze, and R. Rue. Mixing properties of the Swendsen-Wang process on the complete graph and narrow grids. *Journal of Mathematical Physics*, 41(3):1499–1527, 2000. doi:10.1063/1.533194.
7. N. Destainville. Bounding spectral gaps of markov chains: a novel exact multi-decomposition technique. *Journal of Physics A: Mathematical and General*, 36(13):3647–3670, March 2003. doi:10.1088/0305-4470/36/13/301.
8. P. Diaconis and L. Saloff-Coste. Comparison techniques for random walks on finite groups. *The Annals of Applied Probability*, 21:2131–2156, 1993.
9. P. Diaconis and L. Saloff-Coste. Comparison theorems for reversible Markov chains. *Annals of Applied Probability*, 3:696–730, 1993.
10. P. Diaconis and M. Shahshahani. Generating a random permutation with random transpositions. *Z. Wahrscheinlichkeitstheorie Verw. Gebiete*, 57:159–179, 1981.
11. J. Ding, E. Lubetzky, and Yuval Peres. Mixing time of critical Ising model on trees is polynomial in the height. *Communications in Mathematical Physics*, 295(1):161–207, April 2010.
12. P.L. Erdős, I. Miklós, and Z. Toroczkai. New classes of degree sequences with fast mixing swap Markov chain sampling. *Combinatorics Probability and Computing*, 27(2):186–207, 2018.
13. J. Fill. Background on the gap problem. *Unpublished manuscript*, 2003.

14 J. Fill. An interesting spectral gap problem. *Unpublished manuscript*, 2003.

15 R. Grone, K. Heinz Hoffmann, and P. Salamon. An interlacing theorem for reversible Markov chains. *Journal of Physics A: Mathematical and Theoretical*, 41(21):212002, May 2008. doi:10.1088/1751-8113/41/21/212002.

16 S. Haddadan and P. Winkler. Mixing of permutations by biased transposition. In *34th Symposium on Theoretical Aspects of Computer Science*, STACS '17, pages 41:1–41:13, 2017.

17 J. H. Hester and D. S. Hirschberg. Self-organizing linear search. *ACM Computing Surveys (CSUR)*, 17 Issue 3:295–311, 1985.

18 M. Jerrum, J. Son, P. Tetali, and E. Vigoda. Elementary bounds on Poincaré and log-Sobolev constants for decomposable Markov chains. *Ann. Appl. Probab.*, 14(4):1741–1765, November 2004. doi:10.1214/105051604000000639.

19 Donald K. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms. Addison-Wesley, 1969.

20 D. A. Levin, Y. Peres, and E. L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006.

21 N. Madras and D. Randall. Markov chain decomposition for convergence rate analysis. *Annals of Applied Probability*, pages 581–606, 2002.

22 Y. Mao and Liang H. Spectral gap for open Jackson networks. *Acta Mathematica Sinica, English Series*, 31:1879–1894, December 2015. doi:10.1007/s10114-015-4138-3.

23 R. Martin and D. Randall. Sampling adsorbing staircase walks using a new Markov chain decomposition method. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science*, pages 492–502, 2000.

24 S. Miracle, D. Randall, and A. P. Streib. Cluster algorithms for discrete models of colloids with bars. In *8th Workshop on Analytic Algorithmics and Combinatorics*, ANALCO, pages 135–150, 2011.

25 S. Miracle and A.P. Streib. Rapid mixing of k-class biased permutations. In *LATIN*, volume 10807 of *Lecture Notes in Computer Science*, pages 820–834. Springer, 2018.

26 N. Pillai and A. Smith. Elementary bounds on mixing times for decomposable Markov chains. *Stochastic Processes and their Applications*, 127(9):3068–3109, 2017. doi:10.1016/j.spa.2017.02.002.

27 D. Randall. Decomposition methods and sampling circuits in the Cartesian lattice. In *Proceedings of the 26th International Symposium on Mathematical Foundations of Computer Science*, MFCS '01, pages 74–86, London, UK, UK, 2001. Springer-Verlag. URL: http://dl.acm.org/citation.cfm?id=645730.668026.

28 D. Randall and P. Tetali. Analyzing Glauber dynamics by comparison of Markov chains. *Journal of Mathematical Physics*, 41:1598–1615, 2000.

29 O. Reingold, S. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, FOCS '00, pages 3–13, Washington, DC, USA, 2000. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=795666.796583.

30 R. Rivest. On self-organizing sequential search heuristics. *Commun. ACM*, 19(2):63–67, February 1976. doi:10.1145/359997.360000.

31 S. Vadhan. Pseudorandomness. *Foundations and Trends® in Theoretical Computer Science*, 7(1–3):1–336, 2012. doi:10.1561/0400000010.

32 D. Wilson. Mixing times of lozenge tiling and card shuffling Markov chains. *The Annals of Applied Probability*, 1:274–325, 2004.

A Proofs of the decomposition theorems

In this section, we provide illustrate the main ideas of the proofs of the decomposition theorems. Some details are deferred to the full version of the paper. First, in Section A.1, we introduce some notation and terminology which will be useful for the proofs. Several parts

are common to all of the proofs, so we present those parts in Section A.2. In Section A.3, we prove our classical decomposition theorem, Theorem 11. Finally, in Section A.4, we prove our complementary decomposition theorems, Theorems 2, 9, and 10.

A.1 Preliminaries

We first fix some notation and terminology. We write I_n to mean the $n \times n$ identity matrix. The symbol $\otimes$ is used for tensor product. We write $(v)_i$ to mean the i^{th} coordinate of a vector v. The second largest eigenvalue of P_i will be denoted λ_i, and $\lambda_{\max} = \max_i \lambda_i$. The "top eigenvector" of a matrix will be the eigenvector corresponding to the eigenvalue of largest absolute value.

Define $\bar{P}$ to be the aggregated transition matrix on the state space $[r]$ defined by $\bar{P}(i,j) = \pi(\Omega_i)^{-1} \sum_{\sigma\in\Omega_i, \tau\in\Omega_j} \pi(\sigma)P(\sigma,\tau)$. Then $\bar{P}$ is the transition matrix of a reversible Markov chain $\bar{\mathcal{M}}$ with stationary distribution $\bar{\pi}$ defined by $\bar{\pi}(i) := \pi(\Omega_i)$. We call $\bar{\mathcal{M}}$ the *projection* chain.

It is useful to decompose the matrix P into the part that performs restriction moves and the part that performs all other moves. Define $\hat{P}$ as the block diagonal $|\Omega| \times |\Omega|$ matrix with the P_i matrices along the diagonal; i.e. $\hat{P}$ is obtained from P by rejecting moves between different parts of the partition. Define $\tilde{P}$ to be the transition matrix of the Markov chain defined by rejecting moves from σ to τ if σ and τ are within the same Ω_i. Then $(\hat{P}+\tilde{P})(\sigma,\tau) = P(\sigma,\tau)$ unless $\sigma = \tau$, and $(\hat{P}+\tilde{P})(\sigma,\sigma) = P(\sigma,\sigma)+1$, since each move of P gets rejected in exactly one of $\hat{P}$ or $\tilde{P}$ (and of course the probability of transitioning from a state is 1). Therefore, we have $P = \hat{P} + \tilde{P} - I_{|\Omega|}$.

Note that for any pair $\sigma, \tau \in \Omega$, the transitions (σ,τ) and (τ,σ) are either both nonzero in $\tilde{P}$ or both zero in $\tilde{P}$. Thus $\tilde{P}$ is itself the disjoint union of a set of ergodic, reversible Markov chains $\tilde{P}_1, \tilde{P}_2, \ldots, \tilde{P}_{\tilde{r}}$ on state spaces $\tilde{\Omega}_1, \tilde{\Omega}_2, \ldots, \tilde{\Omega}_{\tilde{r}}$. We call these chains *complementary restrictions*.

In order to prove our decomposition results, we wish to apply Equation 3 to P. However, since P may not be symmetric, we appeal to the following symmetrization technique that appears in [20, p. 153]. Given P with stationary distribution π, define a matrix $A := A(P)$ by $A(\sigma,\tau) := \pi(\sigma)^{1/2}\pi(\tau)^{-1/2}P(\sigma,\tau)$. A is similar to P (i.e. they have the same eigenvalues), but is symmetric, so we can infer a bound on the second eigenvalue of P by applying Equation 3 to A. It is easy to check that the top eigenvector of A is $\sqrt{\pi}$, which is the vector with entries $\sqrt{\pi(\sigma)}$ for any $\sigma \in \Omega$.

We apply this same symmetrization technique to other matrices as well. For $i \in [r]$ we let $A_i := A(P_i)$ and for $i \in [\tilde{r}]$ we let $\tilde{A}_i := A(\tilde{P}_i)$. We then write $\hat{A}$ to mean the $|\Omega| \times |\Omega|$ matrix with $\hat{A}(\sigma,\tau) = A_i(\sigma,\tau)$ if $\sigma,\tau \in \Omega_i$ for some $i \in [r]$, and zero otherwise. Analogously, we write $\tilde{A}$ to mean the $|\Omega| \times |\Omega|$ matrix with $\tilde{A}(\sigma,\tau) = \tilde{A}_i(\sigma,\tau)$ if $\sigma,\tau \in \tilde{\Omega}_i$ for some $i \in [\tilde{r}]$, and zero otherwise. It is important to note that $\hat{A} \neq A(\hat{P})$ and $\tilde{A} \neq A(\tilde{P})$. This allows us to write $A = \hat{A} + \tilde{A} - I_{|\Omega|}$.

▶ **Proposition 14.** *The matrix A satisfies $A = \hat{A} + \tilde{A} - I_{|\Omega|}$.*

Let $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_{|\Omega|}$ be the eigenvalues of $\tilde{A}$ with corresponding eigenvectors $v_1, v_2, \ldots, v_{|\Omega|}$. As $\tilde{A}$ is symmetric, the real spectral theorem tells us that its eigenvectors form an orthonormal basis of $\mathbb{R}^{|\Omega|}$. We consider the basis representations $x^{\perp} = \sum_i a_i^{\perp} v_i$ and $x^{\parallel} = \sum_i a_i^{\parallel} v_i$. More generally, for any $v \in \mathbb{R}^{|\Omega|}$, we write $v = \sum_i a_i v_i$ for some constants $a_1, a_2, \ldots, a_{|\Omega|} \in \mathbb{R}$. Also, $\|v\|^2 = \sum_i a_i^2 \|v_i\|^2$, and $v\tilde{A} = \sum_i a_i \mu_i v_i$.

A.2 Key ideas and lemmas for the proofs

We wish to apply Equation 3 to A. Recall that $\sqrt{\pi}$ is the top eigenvector of A. Let $x \in \mathbb{R}^{|\Omega|}$ with $x \perp \sqrt{\pi}$ and $\|x\| = 1$. We will decompose x into two vectors $x^{\|}$ and $x^{\perp}$ as follows (note: this is similar to the vector decomposition used for the Zig Zag Product in [29]). For any $i \in [r]$, let $x_i \in \mathbb{R}^{|\Omega_i|}$ be the vector defined by $x_i(\sigma) = x(\sigma)$ for all $\sigma \in \Omega_i$. Then $x = \sum_i e_i \otimes x_i$. We further decompose x_i into $x_i^{\|}$, the part that is parallel to $\sqrt{\pi_i}$, and $x_i^{\perp}$, the part that is perpendicular to $\sqrt{\pi_i}$; recall that $\sqrt{\pi_i}$ is the top eigenvector of A_i. Finally, define $x^{\|}, x^{\perp} \in \mathbb{R}^{|\Omega|}$ by $x^{\|} = \sum_i e_i \otimes x_i^{\|}$ and $x^{\perp} = \sum_i e_i \otimes x_i^{\perp}$. Hence $x = \sum_i e_i \otimes x_i = x^{\|} + x^{\perp}$. Define $\tilde{x}^{\|}$ and $\tilde{x}^{\perp}$ analogously.

As described in Section 5.3, we will bound $\langle x, xA\rangle$ via Equation 4:

$$\langle x, xA\rangle = \langle x^{\perp}, x^{\perp}A\rangle + \langle x^{\|}, x^{\|}A\rangle + 2\langle x^{\perp}, x^{\|}A\rangle.$$

We need the following simple proposition.

▶ **Lemma 15.** *The following holds:* $x^{\|}A = x^{\|}\tilde{A}$.

Applying Lemma 15, Equation 4 becomes

$$\langle x, xA\rangle = \langle x^{\|}, x^{\|}\tilde{A}\rangle + 2\langle x^{\perp}, x^{\|}\tilde{A}\rangle + \langle x^{\perp}, x^{\perp}(\hat{A} + \tilde{A} - I_{|\Omega|})\rangle.$$

For ease of notation, we define the following quantities:

$$\delta_1 = \frac{\langle x^{\perp}, x^{\perp}\hat{A}\rangle}{\|x^{\perp}\|^2},\ \delta_2 = \frac{\langle x^{\perp}, x^{\perp}\tilde{A}\rangle}{\|x^{\perp}\|^2},\ \delta_3 = \frac{\langle x^{\|}, x^{\|}\hat{A}\rangle}{\|x^{\|}\|^2},\ \delta_4 = \frac{\langle x^{\|}, x^{\|}\tilde{A}\rangle}{\|x^{\|}\|^2}.$$

Plugging these in, we have

$$\langle x, xA\rangle = \delta_4\|x^{\|}\|^2 + 2\langle x^{\perp}, x^{\|}\tilde{A}\rangle + (\delta_1 + \delta_2 - 1)\|x^{\perp}\|^2. \tag{5}$$

Bounding δ_1 and δ_4 is straightforward, and borrows many of the ideas from classical decomposition results. If $x^{\|}\tilde{A}$ were orthogonal to $x^{\perp}$, then doing so would be sufficient to proving a strong decomposition theorem. However, this is not true in general, so we must also bound $\langle x^{\perp}, x^{\|}\tilde{A}\rangle$. Our two types of theorems do so in different ways, which are presented in Sections A.3 and A.4.

The next lemma makes concrete the intuition that if a particular distribution is far from stationarity, then it will either be far from stationarity on some restriction – in which case $\hat{A}$ will bring it closer to stationarity (as in part 1) – or on the projection – in which case $\tilde{A}$ will bring it closer to stationarity (as in part 2). The proof is straightforward from the definitions.

▶ **Lemma 16.** *With the above notation,*
1. $\delta_1 \le \lambda_{\max}$.
2. $\delta_4 \le \bar{\lambda}$.

A.3 Classical decomposition theorems

In this section, we will prove Theorem 17, which is a generalization of Theorem 11.

▶ **Theorem 17.** *Let* $\rho = \sqrt{(1-\delta_2)/\bar{\gamma}}$. *Then* $\gamma(\mathcal{M}) \ge \min_{p^2+q^2=1} \gamma_{\min} q^2 + \bar{\gamma}\left(q\rho - p\right)^2$.

With the technology developed in Section A.2, there is one critical piece remaining to prove Theorem 17, which is to bound the cross terms generated by applying the matrix $\tilde{A}$ to $x^{\|}$.

▶ **Lemma 18.** *With the above notation,* $|\langle x^\perp, x^\parallel \tilde{A}\rangle| \leq \sqrt{(1-\delta_4)(1-\delta_2)}\|x^\parallel\|\|x^\perp\|$.

The proof appears in the full version.

Finally, we are ready to prove Theorem 17.

Proof of Theorem 17. Let $x \in \mathbb{R}^{|\Omega|}$ with $x \perp \sqrt{\pi}$ and $\|x\| = 1$. By Equation 3, $\gamma(\mathcal{M}) \geq 1 - \langle x, xA\rangle$. Applying Lemma 15 and the definitions of δ_1, δ_4, and δ_2, Equation 4 becomes

$$\begin{aligned}\langle x, xA\rangle &= \langle x^\parallel, x^\parallel \tilde{A}\rangle + 2\langle x^\perp, x^\parallel \tilde{A}\rangle + \langle x^\perp, x^\perp(\hat{A} + \tilde{A} - I_{|\Omega|})\rangle \\ &= \delta_4\|x^\parallel\|^2 + 2\langle x^\perp, x^\parallel \tilde{A}\rangle + (\delta_1 + \delta_2 - 1)\|x^\perp\|^2. \end{aligned} \tag{6}$$

Applying Lemma 18, we have

$$\gamma(\mathcal{M}) \geq 1 - (\delta_4\|x^\parallel\|^2 + 2\sqrt{(1-\delta_4)(1-\delta_2)}\|x^\parallel\|\|x^\perp\| + (\delta_1 + \delta_2 - 1)\|x^\perp\|^2).$$

Rearranging terms and using the fact that $1 = \|x\|^2 = \|x^\perp\|^2 + \|x^\parallel\|^2$, we have

$$\gamma(\mathcal{M}) \geq \min_{x^\perp \sqrt{\pi}:\|x\|=1} (1-\delta_1)\|x^\perp\|^2 + \left(\sqrt{1-\delta_2}\|x^\perp\| - \sqrt{1-\delta_4}\|x^\parallel\|\right)^2. \tag{7}$$

By setting $q = \|x^\perp\|$ and $p = \|x^\parallel\|$, we immediately get

$$\gamma(\mathcal{M}) \geq \min_{p^2+q^2=1} (1-\delta_1)q^2 + \left(\sqrt{1-\delta_2}q - \sqrt{1-\delta_4}p\right)^2.$$

Using a bit of calculus one may show that the expression on the right is minimized when $(1-\delta_1)$ is minimized, when $(1-\delta_2)$ is maximized and when $(1-\delta_4)$ is minimized. By Lemma 16, $(1-\delta_1) \geq \gamma_{\min}$ and $(1-\delta_2) \geq \bar{\gamma}$. Therefore, we have

$$\gamma(\mathcal{M}) \geq \min_{p^2+q^2=1} \gamma_{\min}q^2 + \left(q\sqrt{1-\delta_2} - p\sqrt{\bar{\gamma}}\right)^2. \qquad ◀$$

The statement of Theorem 17 is admittedly technical. However, from it we may derive several corollaries, as listed in Section 5. It is simple to show that Theorem 11 follows from Theorem 17 by noticing that $\delta_2 \geq 1 - 2T$.

We do not currently have a comparison between our Theorem 11 and Corollary 2 of [18], which requires a pointwise bound of π_i^j. However, see Remark 19 which shows that their result would not be sufficient for our application to permutations.

▶ **Remark 19.** It is worth pointing out that the decomposition of $\mathcal{M}_{\sigma_{i-1}}$ that we described in Section 4 does not satisfy the regularity conditions of [18] needed to obtain a better bound. For any $\upsilon \in \Omega_j$, define

$$\pi_j^{j'}(\upsilon) = \pi_j(\upsilon)\frac{\sum_{\upsilon' \in \Omega_{j'}} P(\upsilon, \upsilon')}{\bar{P}(j, j')}.$$

We need to bound $\pi_j^{j'}(\upsilon)/\pi_j(\upsilon)$ for any j, j', and $\upsilon \in \Omega_j$. For example, let $\sigma_2 =$ 12_11111_2_1_ _ and $\sigma_3 =$ 12311111_231_ _. Notice that the two permutations $\upsilon_1 =$ 12311111423156 and $\upsilon_2 =$ 12311111523146 are in the same restriction Ω_j (i.e. they are both consistent with σ_3). They each have a single move to $\Omega_{j'}$: the move of swapping the first 3 with the 4 (in the case of υ_1) or 5 (in the case of υ_2). However, the probability of these moves differ by a factor of $(\overline{q}_{4,3}/\overline{q}_{5,3})(\overline{q}_{4,1}/\overline{q}_{5,1})^5$, as there are five 1's between them. In principle, there could be order n smaller numbers between the two numbers we are swapping. Thus, $\pi_j^{j'}(\upsilon)/\pi_j(\upsilon)$ cannot be uniformly bounded to within $1 \pm \eta$ unless η is exponentially large.

A.4 Complementary decomposition theorems

Next we use the technology developed in Section A.3 to prove Theorem 10. Recall that $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_{|\Omega|}$ are the eigenvalues of $\tilde{A}$ with corresponding eigenvectors $v_1, v_2, \ldots, v_{|\Omega|}$, and that for any $v \in \mathbb{R}^{|\Omega|}$, we write $v = \sum_i a_i v_i$ for some constants $a_1, a_2, \ldots, a_{|\Omega|} \in \mathbb{R}$. Also, $\|v\|^2 = \sum_i a_i^2 \|v_i\|^2$, and $v\tilde{A} = \sum_i a_i \mu_i v_i$. Define the set $S = \{i : \mu_i = 1\}$. Let $\tilde{\delta}_1 = \frac{\langle \tilde{x}^\perp, \tilde{x}^\perp \tilde{A} \rangle}{\|\tilde{x}^\perp\|^2}$. Now we can make an explicit statement about the gap of $\mathcal{M}$; notice the equality in Equation 8.

▶ **Theorem 10.**

$$\gamma(\mathcal{M}) = \min_{x \perp \sqrt{\pi}, \|x\|=1} (1-\delta_1)\|x^\perp\|^2 + (1-\tilde{\delta}_1)\|\tilde{x}^\perp\|^2. \tag{8}$$

In particular,

$$\gamma(\mathcal{M}) \geq \min_{x \perp \sqrt{\pi}, \|x\|=1} \gamma_{\min}\|x^\perp\|^2 + \tilde{\gamma}_{\min}\|\tilde{x}^\perp\|^2. \tag{9}$$

Proof. Notice $\tilde{\delta}_1 \|\tilde{x}^\perp\|^2 = \sum_{i \in S} \mu_i (a_i^\perp + a_i^\parallel)^2$. Thus,

$$(1-\tilde{\delta}_1)\|\tilde{x}^\perp\|^2 = \sum_i (1-\mu_i)(a_i^\perp + a_i^\parallel)^2 = (1-\delta_2)\|x^\perp\|^2 + (1-\delta_4)\|x^\parallel\|^2 - 2\langle x^\perp, x^\parallel \tilde{A}\rangle.$$

On the other hand, from Equation 6, we have

$$1 - \langle x, xA\rangle = (1-\delta_1)\|x^\perp\|^2 + (1-\delta_2)\|x^\perp\|^2 + (1-\delta_4)\|x^\parallel\|^2 - 2\langle x^\perp, x^\parallel \tilde{A}\rangle.$$

Thus, for all $x \perp \sqrt{\pi}$ with norm 1, we have

$$1 - \langle x, xA\rangle = (1-\delta_1)\|x^\perp\|^2 + (1-\tilde{\delta}_1)\|\tilde{x}^\perp\|^2.$$

Applying Equation 3, we get Equation 8. To get Equation 9, we apply Lemma 16, which yields $1 - \delta_1 \geq 1 - \lambda_{\max} = \gamma_{\min}$. An analogous statement to Lemma 16 holds for $\tilde{\delta}_1$, and shows $1 - \tilde{\delta}_1 \geq \tilde{\gamma}_{\min}$. ◀

It remains to prove Theorem 9. By Theorem 10, if $\gamma_{\min}$ and $\tilde{\gamma}_{\min}$ are not too small, it suffices to show that $\|x^\perp\|^2$ and $\|\tilde{x}^\perp\|^2$ cannot both be small. To this end, we further decompose $x^\perp$ and $\tilde{x}^\parallel$ based on the eigenvectors of $\tilde{A}$. Define $S = \{i : \mu_i = 1\}$ and vectors $x_{11} = \sum_{i \in S} a_i^\parallel v_i$ and $x_{12} = \sum_{i \notin S} a_i^\parallel v_i$. Similarly, let $x_{21} = \sum_{i \in S} a_i^\perp v_i$ and $x_{22} = \sum_{i \notin S} a_i^\perp v_i$. Notice $\tilde{x}^\parallel = x_{11} + x_{21}$ and $\tilde{x}^\perp = x_{12} + x_{22}$, so that the vectors in each row (respectively, column) of the following table sum to the vector in its row (respectively, column) label.

	$\tilde{x}^\parallel$	$\tilde{x}^\perp$
$x^\parallel$	x_{11}	x_{12}
$x^\perp$	x_{21}	x_{22}

The vectors within each row are orthogonal, as they are in the span of eigenvectors with distinct eigenvalues. However, the vectors within each column are not necessarily orthogonal.

The idea of the proof of Theorem 9 is that if $\|x_{11}\|$ is small, then $\|x^\perp\|^2 + \|\tilde{x}^\perp\|^2$ is large. The following lemma states that ϵ-orthogonality is sufficient to guarantee $\|x_{11}\|$ is small.

▶ **Lemma 20.** *Let ϵ be as defined in Equation 1. Then $\|x_{11}\|^2 \leq \epsilon$.*

Proof. Recall x_{11} is the projection of $x^{\|}$ onto the top eigenvectors of $\tilde{A}$. The top eigenvectors of $\tilde{A}$ are precisely the set of all $\sqrt{\tilde{\pi}_j}$ for $j \in [\tilde{r}]$. Therefore,

$$x_{11} = \sum_j \frac{\langle x^{\|}, \sqrt{\tilde{\pi}_j}\rangle}{\|\sqrt{\tilde{\pi}_j}\|^2}\sqrt{\tilde{\pi}_j}.$$

As the eigenvectors of $\tilde{A}$ are an orthonormal basis, we have

$$\|x_{11}\|^2 = \sum_j \langle x^{\|}, \sqrt{\tilde{\pi}_j}\rangle^2.$$

For any $j \neq j' \in [\tilde{r}]$ and any $\sigma \in \tilde{\Omega}_{j'}$, $\sqrt{\tilde{\pi}_j(\sigma)} = 0$ and for $i \in [r]$, $\tilde{\pi}_j(\Omega_i \cap \tilde{\Omega}_j) = \pi(\Omega_i \cap \tilde{\Omega}_j)/\pi(\tilde{\Omega}_j)$. Therefore,

$$\langle x^{\|}, \sqrt{\tilde{\pi}_j}\rangle = \sum_i \sum_{\sigma\in\Omega_i} x^{\|}(\sigma)\sqrt{\tilde{\pi}_j(\sigma)} = \sum_i \alpha_i \sum_{\sigma\in\Omega_i\cap\tilde{\Omega}_j} \sqrt{\pi_i(\sigma)\tilde{\pi}_j(\sigma)} = \sum_{i\in[r]} \alpha_i \frac{\pi(\Omega_i\cap\tilde{\Omega}_j)}{\sqrt{\pi(\Omega_i)\pi(\tilde{\Omega}_j)}}. \tag{10}$$

Since $x \perp \sqrt{\pi}$ and $x^{\perp} \perp \sqrt{\pi}$ by definition, it follows that $x^{\|} \perp \sqrt{\pi}$ as well. This implies that $\alpha \perp \sqrt{\tilde{\pi}}$, as

$$0 = \langle x^{\|}, \sqrt{\pi}\rangle = \sum_i \alpha_i \sum_{\sigma\in\Omega_i} \sqrt{\pi_i(\sigma)\pi(\sigma)} = \sum_i \alpha_i \sum_{\sigma\in\Omega_i} \frac{\sqrt{\pi(\sigma)}}{\sqrt{\pi(\Omega_i)}}\sqrt{\pi(\sigma)} = \sum_i \alpha_i \sqrt{\pi(\Omega_i)}, \tag{11}$$

and this final term is equal to $\sum_i \alpha_i\sqrt{\tilde{\pi}_i} = \langle \alpha, \sqrt{\tilde{\pi}}\rangle$. Multiplying Equation 11 by $\pi(\tilde{\Omega}_j)$ and subtracting it from Equation 10, we have

$$\langle x^{\|}, \sqrt{\tilde{\pi}_j}\rangle = \sum_{i\in[r]} \alpha_i \left(\frac{\pi(\Omega_i\cap\tilde{\Omega}_j)}{\sqrt{\pi(\Omega_i)\pi(\tilde{\Omega}_j)}} - \sqrt{\pi(\Omega_i)\pi(\tilde{\Omega}_j)} \right) = \langle \alpha, V_j\rangle,$$

where

$$V_j(i) := \left(\frac{\pi(\Omega_i\cap\tilde{\Omega}_j)}{\sqrt{\pi(\Omega_i)\pi(\tilde{\Omega}_j)}} - \sqrt{\pi(\Omega_i)\pi(\tilde{\Omega}_j)} \right) = \sqrt{\pi(\Omega_i\cap\tilde{\Omega}_j)}(\sqrt{r(i,j)} - 1/\sqrt{r(i,j)}).$$

By the Cauchy-Schwartz inequality and the fact that $\|\alpha\| = \|x^{\|}\| \leq \|x\| = 1$, we have $\langle \alpha, V_j\rangle \leq \|\alpha\|\|V_j\| = \|V_j\|$. Therefore we get,

$$\|x_{11}\|^2 = \sum_j \langle x^{\|}, \sqrt{\tilde{\pi}_j}\rangle^2 \leq \sum_j \|V_j\|^2 = \sum_{i,j} \pi(\Omega_i\cap\tilde{\Omega}_j)(\sqrt{r(i,j)} - 1/\sqrt{r(i,j)})^2. \quad \blacktriangleleft$$

To prove Theorem 9 from Theorem 10, we must show that if $\|x_{11}\|^2 \leq \epsilon$, then $\|x^{\perp}\|^2 + \|\tilde{x}^{\perp}\|^2 \geq (1-\sqrt{\epsilon})^2$. As the sum of the squared norms of the vectors in the above table is 1, it is reasonable to expect that if $\|x_{11}\|^2$ is small, then $\|x^{\perp}\|^2 + \|\tilde{x}^{\perp}\|^2$ is large. However, this is not as straightforward as one might expect, as the vectors within each column are not necessarily orthogonal, so we may have $\|\tilde{x}^{\perp}\|^2 < \|x_{12}\|^2 + \|x_{22}\|^2$. The proof is deferred to the full version of this paper.

Improved Explicit Hitting-Sets for ROABPs

Zeyu Guo[1]
Department of Computer Science, University of Haifa, Israel
zguotcs@gmail.com

Rohit Gurjar
Department of Computer Science and Engineering, IIT Bombay, India
rohitgurjar0@gmail.com

Abstract

We give improved explicit constructions of hitting-sets for *read-once oblivious algebraic branching programs* (ROABPs) and related models. For ROABPs in an unknown variable order, our hitting-set has size polynomial in $(nr)^{\frac{\log n}{\max\{1,\log\log n-\log\log r\}}}d$ over a field whose characteristic is zero or large enough, where n is the number of variables, d is the individual degree, and r is the width of the ROABP. A similar improved construction works over fields of arbitrary characteristic with a weaker size bound.

Based on a result of Bisht and Saxena (2020), we also give an improved explicit construction of hitting-sets for *sum of several ROABPs*. In particular, when the characteristic of the field is zero or large enough, we give polynomial-size explicit hitting-sets for sum of constantly many log-variate ROABPs of width $r = 2^{O(\log d/\log\log d)}$.

Finally, we give improved explicit hitting-sets for polynomials computable by width-r ROABPs in any variable order, also known as *any-order ROABPs*. Our hitting-set has polynomial size for width r up to $2^{O(\log(nd)/\log\log(nd))}$ or $2^{O(\log^{1-\epsilon}(nd))}$, depending on the characteristic of the field. Previously, explicit hitting-sets of polynomial size are unknown for $r = \omega(1)$.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases polynomial identity testing, hitting-set, ROABP, arithmetic branching programs, derandomization

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.4

Category RANDOM

Acknowledgements We thank Nitin Saxena, Sumanta Ghosh, and Pranav Bisht for many helpful discussions about PIT for ROABPs and related models.

1 Introduction

Polynomial identity testing (PIT) is one of the fundamental problems in the area of derandomization. The problem asks whether a given multi-variate polynomial is identically zero. For example, the polynomial $(x+y)(x-y)-x^2-y^2$ is identically zero. The input to the problem can be given as an algebraic formula or circuit or other algebraic computation models like arithmetic branching programs or determinant of a symbolic matrix. The problem is not known to be polynomial-time solvable. One way to test zeroness could be to check whether the coefficient of each monomial is zero in the polynomial. However, for a given circuit or branching program, it might take exponential time (in the input size) to compute coefficients.

On the other hand, there is a simple (polynomial time) randomized algorithm to test zeroness of a given polynomial: just evaluate the input circuit at a random point and see if the evaluation is nonzero. It is known that a nonzero polynomial evaluated at a random

[1] Part of this work was done while the first author was a postdoc at IIT Kanpur.

© Zeyu Guo and Rohit Gurjar;
licensed under Creative Commons License CC-BY

point from gives a nonzero value with high probability [14, 6, 22, 20]. More precisely, for an n-variate polynomial of degree d, if you evaluate it at a random point from S^n for some subset $S \subseteq \mathbb{F}$, then the probability of the evaluation being zero is at most $d/|S|$. The polynomial identity testing question can be asked over any field, however as this randomized algorithm suggests, in case of finite characteristic we need to take a large enough field extension.

To obtain a deterministic polynomial time algorithm for the polynomial identity testing has been a long open question. Such an algorithm is known only for some special cases, for example, read-once oblivious arithmetic branching programs (ROABP) (for more such cases, see [21, 18, 19]). Deterministic identity testing for ROABPs has been widely studied in the last decade. One reason for such an interest in this special case is that it can be considered as an algebraic analogue of the RL vs. L question. An ROABP is a product of matrices $f = \beta^\top f_1 f_2 \cdots f_n \gamma$ where $\beta, \gamma \in \mathbb{F}^{r\times 1}$ and $f_i \in \mathbb{F}^{r\times r}[x_{\pi(i)}]$ is a matrix with entries being polynomials in the variable $x_{\pi(i)}$ for each $1 \leq i \leq n$ for some permutation $\pi\colon [n] \to [n]$. The permutation π is said to be the variable order of the ROABP.

Raz and Shpilka [16] gave the first polynomial time algorithm to test whether a given ROABP computes a nonzero polynomial. PIT is also studied in the so-called black-box model, where one does not have access to the circuit/ABP computing the polynomial. Instead, one has to construct an explicit hitting-set – a set of points with the guarantee that every nonzero polynomial in the class of interest gives a nonzero evaluation on at least one of the points in the set. Here, by *explicit* we mean that every point in the hitting-set should be computable in polynomial time. Forbes and Shpilka [9] first gave a quasi-polynomial size explicit hitting-set for ROABPs, when the variable order is known. In subsequent works [8, 1], a quasi-polynomial size explicit hitting-set was also constructed for the unknown order case. Constructing a polynomial-size explicit hitting-set for ROABPs remains a challenging open question. This situation is somewhat similar to that for pseudorandom generators (PRG) for log-space computation. There are no PRGs known with the optimal seed length, i.e. $O(\log n)$, but are known with close to optimal seed length i.e., $O(\log^2 n)$ [13, 12, 15].

There has been a sequence of work in last few years which improve the hitting-set construction for ROABPs with respect to various parameters. There are usually three parameters associated with ROABPs, its length or depth n, which is same as the number of variables, the individual degree d – maximum degree of any variable, and the width r – the size of the matrices involved in the product. The hitting-set of [9] and of [1] both had size $(ndr)^{O(\log n)}$, for the cases of known and unknown variable orders, respectively. For the known order case slightly better results are known. The first paper [9] also gave a bound of $(ndr)^{O(\log n/\max\{1,\log\log n - \log\log r\})}$, which is better when the width r is relatively small. For the small width case, another improved bound of $ndr^{\log n}$ was obtained by [10], when the field characteristic is zero or large enough.

A special class of polynomials, which is known to have better hitting-sets, is called any-order ROABPs. These are polynomials that have small-width ROABPs in every possible variable order. Any-order ROABPs generalize commutative ROABPs[2] and diagonal circuits [17]. Building upon the techniques of [8], an explicit hitting-set of size $(ndr)^{O(\log\log r)}$ for any-order ROABPs was obtained in [10].

A more general model, namely, sum of constantly many ROABPs was considered by [11]. As is known for ROABPs, they could give a polynomial time algorithm for sum of constantly many ROABPs in the white-box case and also a quasi-polynomial size explicit hitting-set. More precisely, for a sum of c ROABPs, their hitting-set size is $(ndr)^{O(c\cdot 2^c \log(ndr))}$.

[2] We say an ROABP is commutative if its output does not change under any permutation of the matrices involved in the product. The usage of "commutative ROABP" is slightly different in [8], which actually refers to any-order ROABPs in this paper.

Recently, Bisht and Saxena [3] considered PIT for ROABP and sum of ROABPs in the small variate regime. For a sum of c ROABPs, they gave a hitting-set of size $\text{poly}(r^{n3^c}, d^c)$, which also means a hitting-set of size $\text{poly}(r^n, d)$ for an ROABP. These results are better than those of [11] and [1] respectively, when $n = O(\log(rd))$ and $r = O(1)$.

In this work we give improved explicit hitting-sets for ROABPs (unknown order), sum of several ROABPs (small variate regime) and any-order ROABPs with respect to various parameters. Though, we are still away from a polynomial size hitting-set, one important feature of our results is a better dependence on the degree parameter d. In particular, for unknown order and any-order ROABPs, our dependence on d is only polynomial instead of quasi-polynomial (when field characteristic is zero/large). This is somewhat analogous to a recent result for read-once boolean branching programs [4], where they construct a hitting-set of size quasi-polynomial in length and width, but the dependence on the error parameter $1/\epsilon$ is nearly-polynomial.

1.1 Our Results

We now state our main theorems, which give improved explicit constructions of hitting-sets for ROABPs in an unknown order, sum of several ROABPs, and any-order ROABPs.

1.1.1 ROABPs in an Unknown Order

We have the following result for general ROABPs in an unknown order.

▶ **Theorem 1.** *Let $\mathcal{C}$ be the family of polynomials $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ computed by ROABPs of length n, width r and individual degree d in an unknown order. If $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > d$, then there exists an explicit hitting-set for $\mathcal{C}$ of size polynomial in*

$$M(n, r, d) := d \cdot (nr)^{\frac{\log n}{\max\{1, \log\log n - \log\log r\}}}.$$

In arbitrary characteristic, there exists an explicit hitting-set for $\mathcal{C}$ of size polynomial in

$$M'(n, r, d) := \begin{cases} (nr)^{\log n} d & nd \leq r^2, \\ (nd)^{\frac{\log n}{\log\log(nd) - \log\log r}} & r^2 < nd < r^n, \\ nd & nd \geq r^n. \end{cases}$$

Comparison with Previous Work. In all cases, our bounds are strictly better than the previous best bound of $(ndr)^{O(\log n)}$ [1] for unknown order ROABPs. In particular, our dependence on the individual degree d is better. Our bounds are also better than known order case results of [9]. Recall that they had an explicit hitting-set of size $(ndr)^{O(\log n)}$, and for small r, they had an explicit hitting-set of size $(ndr)^{O(\log n/\max\{1, \log\log n - \log\log r\})}$ (not explicitly written, but follows from [9, Theorem 3.24]). These results are subsumed by Theorem 1. In fact, we follow the same idea in [9] of merging $k \geq 2$ parts of the ROABP at each level of the recursion. We note our construction has two advantages compared with [9]:

- Theorem 1 applies to ROABPs in an *unknown* order, while it is not clear how to achieve the same using the construction in [9]. The requirement that the hitting-set works in an unknown order is crucial for the model of the sum of several ROABPs which is discussed below.
- When $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > d$, our size bound depends only polynomially on the individual degree bound d, which gives much smaller hitting-sets compared with [9] if $n, r \ll d$.

The hitting-set constructed in [3, Lemma 9] for $\mathcal{C}$ of size $\text{poly}(r^n, d)$ is also subsumed by our result. When $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > d$, Theorem 1 improves this by giving an explicit hitting-set of size $M(n, r, d) \leq \text{poly}((nr)^{\log n}, d)$. In particular, in the log-variate case $n = O(\log(rd))$ considered in [3], they can achieve a $\text{poly}(n, r, d)$-size hitting-set only when $r = O(1)$, while we can achieve the same for r up to $2^{O(\log d/ \log\log d)}$. In arbitrary characteristic, we obtain a worse size bound $M'(n, r, d)$, which still subsumes [3, Lemma 9].[3]

Finally, in comparison with the hitting-set of [10, Theorem 3.6] of size $\text{poly}(n^{\log r}, d)$ for *known* order ROABPs, our bound of $M(n, r, d)$ is weaker. In particular, they give polynomial-size hitting-sets when the width r is constant. However, their result is not known to be extendible to ROABPs in an unknown order.

1.1.2 Sum of Several ROABPs

The paper [3] studied the problem of constructing hitting-sets for the sum of several (log-variate) ROABPs and established a reduction from this problem to constructing hitting-sets for ROABPs in an unknown order. Using this reduction, we obtain the following result for the model of sum of several ROABPs.

▶ **Theorem 2.** *Let $\mathcal{C}$ be the family of polynomials $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ computed by the sum of c ROABPs of length n, width $r \geq 2$ and individual degree d in unknown and possibly different orders.*

1. *If $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > d$, there exists an explicit hitting-set for $\mathcal{C}$ of size polynomial in $2^{cn} \cdot M(n, (2r)^{3^c}, d)^c$ where $M(\cdot, \cdot, \cdot)$ is as in Theorem 1. In particular, the hitting-set has size $\text{poly}(d)$ when $c = O(1)$, $n = O(\log d)$ and $r = 2^{O(\log d/ \log\log d)}$.*
2. *In arbitrary characteristic, there exists an explicit hitting-set for $\mathcal{C}$ of size polynomial in $2^{cn} \cdot M'(n, (2r)^{3^c}, d)^c$ where $M'(\cdot, \cdot, \cdot)$ is as in Theorem 1.*

The paper [3] constructed an explicit hitting-set of size $\text{poly}(r^{n3^c}, d^c)$ for $\mathcal{C}$, which has size $\text{poly}(d)$ when $n = O(\log d)$ and c, r are constants. This result is subsumed by our Theorem 2 (2) since $M'(n, r, d)$ is bounded by $\text{poly}(n, r^n, d)$. Moreover, when $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > d$, our Theorem 2 (1) yields a $\text{poly}(d)$-size hitting-set for $n = O(\log d)$, $c = O(1)$ and $r = 2^{O(\log d/ \log\log d)}$ (instead of constant r).

1.1.3 Any-Order ROABPs

Recall that any-order ROABPs are polynomials that have small-width ROABPs in every possible variable order. We obtain the following result for any-order ROABPs.

▶ **Theorem 3.** *Let $\mathcal{C}$ be the family of polynomials $f \in \mathbb{F}[x_1, x_2, \ldots, x_n]$ computed by ROABPs of length n, width r and individual degree d in any order.*

1. *If $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > n^4(d+1)^2$, then there exists an explicit hitting-set for $\mathcal{C}$ of size $\text{poly}(n, r^{\log\log r}, d)$. In particular, the hitting-set has size $\text{poly}(n, d)$ for $r = 2^{O(\log(nd)/ \log\log(nd))}$.*
2. *In arbitrary characteristic, there exists an explicit hitting-set for $\mathcal{C}$ of size*

$$\text{poly}(r^{\log\log r}, (nd)^{1+\frac{\log\log r}{\max\{1, \log\log(nd) - \log\log r\}}}).$$

So the hitting-set has size $\text{poly}(n, d)$ for $r = 2^{O(\log^{1-\epsilon}(nd))}$ and any constant $0 < \epsilon < 1$.

---

[3] We note that [3, Lemma 9] is proved using ideas different from ours. To directly see that our bound subsumes the bound $\text{poly}(r^n, d)$ when $r^2 < nd < r^n$, write $nd = r^{n^{1/e}}$ with $1 < e < \log n$ and note $M'(n, r, d) = (nd)^{\frac{\log n}{\log\log(nd) - \log\log r}} = r^{en^{1/e}} = r^{O(n)}$.

The previous best explicit construction of hitting-sets for any-order ROABPs [10] has size $(ndr)^{O(\log\log r)}$, which is superpolynomial for width $r = \omega(1)$. Our hitting-set has polynomial size for r up to $2^{O(\log(nd)/\log\log(nd))}$ or $2^{O(\log^{1-\epsilon}(nd))}$ depending on the characteristic of $\mathbb{F}$.

1.2 Proof Techniques

We prove our results by combining the analyses in previous work [8, 1, 10, 11, 3] with the following ideas.

(1) Low-Degree Concentration via Random Shift. Randomly shifting a multivariate polynomial is an important and common technique in polynomial identity testing for ROABPs and related models. For example, it was used in [2, 8, 1, 10, 11, 7] to achieve rank concentration of polynomials. We use a simple version of this technique, applied only to univariate polynomials: View the layers of a width-r ROABP as univariate polynomials $f_1(x_1), f_2(x_2), \ldots, f_n(x_n)$ with matrix-valued coefficients. We preprocess these polynomials by performing the shift $f_i(x_i) \mapsto f_i(x_i + \alpha)$ simultaneously for $i = 1, 2, \ldots, n$ with randomly chosen $\alpha \in \mathbb{F}$.

Assuming $\mathrm{char}(\mathbb{F})$ is zero or large, a standard argument shows that with high probability, each of the new polynomials $f_i(x_i + \alpha)$ is *low-degree concentrated* in the sense that its coefficient span, which has dimension $\ell_i \leq r^2$, is spanned by the coefficients of the ℓ_i monomials with the lowest degrees. This is useful when the width r is much smaller than the degree bound d of the polynomials, as it allows us to reduce d to r^2 in the analysis.

We remark that a generalization of this technique was developed in [7], where it was shown that a (pseudo-)random shift achieves *low-cone concentration* for multivariate polynomials [7, Theorem 2]. We only need the special case for univariate polynomials, which is classical and uses the nonsingularity of the Wronskian matrix.

(2) Merging Multiple Parts at Each Level of the Recursion. Explicit hitting-sets for ROABPs of size $(ndr)^{O(\log n)}$ were constructed in [9, 1], which may be seen as analogues of the PRG constructions in [13] and [12] for read-once branching programs. Roughly speaking, these hitting-sets are recursively constructed as follows: Divide the ROABP into two parts, construct a hitting-set for each part recursively, and then merge them at the cost of increasing the size by a factor polynomial in ndr. The size of the final hitting-set is $(ndr)^{O(\log n)}$ as the recursion tree has depth $O(\log n)$.

A slightly better construction was also given in [9] for ROABPs of small width. The idea is to merge k parts of the ROABP at each level of the recursion, where k is possibly greater than two. We use the same idea in this paper but replace the construction in [9] by the one in [1], which has the advantage of working for ROABPs in an unknown order. The cost incurred at each level of the recursion is bounded by $\mathrm{poly}(n, d, r^k)$ while there are $O(\log n/\log k)$ levels. When $\mathrm{char}(\mathbb{F})$ is zero or large, the cost incurred at each level may be improved to $\mathrm{poly}(n, r^k)$ by using the idea (1) above. We then choose the optimal k according to the parameters n, r and d.

(3) Reducing the Number of Variables via Hashing. In [8, 10], hitting-sets for any-order ROABPs are constructed in two steps: The first step is to explicitly construct a small set $T \subseteq \mathbb{F}^n$ such that for some $s = (s_1, s_2, \ldots, s_n) \in T$, performing the shift $x_i \mapsto x_i + s_i$ achieves *low-support concentration* for any-order ROABPs. The second step is to convert an any-order ROABP with low-support concentration into a short ROABP.

In [8], the cost of the first step (i.e., the size of T) is polynomial in n and $d^{\log r}$. This was later improved to $(ndr)^{O(\log\log r)}$ in [10]. In this paper, we further improve the cost to $\mathrm{poly}(n, r^{\log\log r}, d)$ when $\mathrm{char}(\mathbb{F})$ is zero or large. In arbitrary characteristic, we obtain a worse bound which still improves those in [8, 10]. See Theorem 16 for details.

One crucial idea used in [8, 10] (which originates from [2]) is that for any-order ROABPs, low-support concentration is a "local" property. Namely, in order to achieve low-support concentration of n-variate any-order ROABPs, it suffices to achieve it when restricting to every subset of ℓ variables, where $\ell = O(\log r)$. In this paper, we use a construction similar to the one in [10] except that we further exploit the locality by using hash functions. This has the effect of reducing n to $\mathrm{poly}(\log r)$ in the analysis, which leads to the improvement.

Organization of the Paper. Preliminaries and notations are given in Section 2. Theorem 1 and 2 are proved in Section 3. Theorem 3 is proved in Section 4. Finally, some open problems are listed in Section 5.

2 Preliminaries

Notations. Let $\mathbb{N} := \{0, 1, 2, \dots\}$ and $\mathbb{N}^+ := \{1, 2, \dots\}$. Denote $\{1, 2, \dots, n\}$ by $[n]$. The cardinality of a set S is denoted by $|S|$. Denote by $\log a$ the logarithm of a with base two.

Let $\mathbb{F}$ be a field. Throughout this paper, we always assume $|\mathbb{F}|$ is large enough. This can be guaranteed by replacing $\mathbb{F}$ with an extension field if necessary. We often write $\mathbf{x}$ as a shorthand for a list of variables $x_1, x_2, \dots, x_n$. For $\mathbf{a} = (a_1, a_2, \dots, a_n) \in \mathbb{N}^n$, write $\mathbf{x}^\mathbf{a}$ for the monomial $\prod_{i=1}^n x_i^{a_i}$. The *support* of $\mathbf{x}^\mathbf{a}$ is $\mathrm{supp}(\mathbf{x}^\mathbf{a}) := \{i \in [n] : a_i > 0\}$. The set of all monomials in $x_1, x_2, \dots, x_n$ is denoted by $\mathcal{M}(x_1, x_2, \dots, x_n)$ or $\mathcal{M}(\mathbf{x})$.

For an algebra $\mathbb{A}$ over $\mathbb{F}$, write $\mathbb{A}[\mathbf{x}]$ for the ring of polynomials in the variables $\mathbf{x}$ with coefficients in $\mathbb{A}$. For $f \in \mathbb{A}[\mathbf{x}]$ and a monomial $m = \mathbf{x}^\mathbf{a}$, denote by $\mathrm{coef}_f(m) \in \mathbb{A}$ the coefficient of m in f. The linear span of a set $T \subseteq \mathbb{A}$ over $\mathbb{F}$ is denoted by $\mathrm{span}\, T$. The *coefficient span* of $f \in \mathbb{A}[\mathbf{x}]$ is $\mathrm{span}(f) := \mathrm{span}\{\mathrm{coef}_f(m) : m \text{ is a monomial in } f\}$.

More generally, for an extension field $\mathbb{K}$ of $\mathbb{F}$, denote by $\mathbb{A} \otimes_\mathbb{F} \mathbb{K}$ the tensor product of $\mathbb{A}$ and $\mathbb{K}$ over $\mathbb{F}$, which is an algebra over $\mathbb{K}$, i.e., $\mathbb{A} \otimes_\mathbb{F} \mathbb{K}$ is obtained from $\mathbb{A}$ by extending the field of scalars from $\mathbb{F}$ to $\mathbb{K}$. For $f \in (\mathbb{A} \otimes_\mathbb{F} \mathbb{K})[\mathbf{x}]$ and a monomial $m = \mathbf{x}^\mathbf{a}$, again denote by $\mathrm{coef}_f(m) \in \mathbb{A} \otimes_\mathbb{F} \mathbb{K}$ the coefficient of m in f. The linear span of a set $T \subseteq \mathbb{A} \otimes_\mathbb{F} \mathbb{K}$ over $\mathbb{K}$ is denoted by $\mathrm{span}_\mathbb{K} T$. The coefficient span of $f \in (\mathbb{A} \otimes_\mathbb{F} \mathbb{K})[\mathbf{x}]$ over $\mathbb{K}$ is $\mathrm{span}_\mathbb{K}(f) := \mathrm{span}_\mathbb{K}\{\mathrm{coef}_f(m) : m \text{ is a monomial in } f\}$.

Let $r \in \mathbb{N}^+$ be a parameter. From now on, we fix $\mathbb{A}$ to be $M_{r\times r}(\mathbb{F})$, the algebra of $r \times r$ matrices over $\mathbb{F}$, even though statements in this paper often hold over other algebras as well. So $\mathbb{A} \otimes_\mathbb{F} \mathbb{K}$ is simply $M_{r\times r}(\mathbb{K})$, the algebra of $r \times r$ matrices over $\mathbb{K}$.

Rank Concentration. We need the following definitions about rank concentration.

▶ **Definition 4.** *Let $f \in \mathbb{A}[\mathbf{x}] = \mathbb{A}[x_1, x_2, \dots, x_n]$ be a polynomial over $\mathbb{A}$. For a set $S \subseteq \mathcal{M}(\mathbf{x})$ of monomials, we say f is* concentrated on S *if* $\mathrm{span}(f) = \mathrm{span}\{\mathrm{coef}_f(m) : m \in S\}$. *For $\ell \in \mathbb{N}$, we say f is* ℓ-support concentrated *if it is concentrated on $S = \{\mathbf{x}^\mathbf{a} : |\mathrm{supp}(\mathbf{x}^\mathbf{a})| < \ell\}$. Similarly, we say f is* ℓ-degree concentrated *if it is concentrated on $S = \{\mathbf{x}^\mathbf{a} : \deg(\mathbf{x}^\mathbf{a}) < \ell\}$.*

More generally, for an extension field $\mathbb{K}$ of $\mathbb{F}$, we say $f \in (\mathbb{A} \otimes_\mathbb{F} \mathbb{K})[\mathbf{x}]$ is concentrated on S over $\mathbb{K}$, ℓ-support concentrated *over $\mathbb{K}$, or* ℓ-degree concentrated *over $\mathbb{K}$ if f satisfies the corresponding property above with* $\mathrm{span}(f)$ *and* $\mathrm{span}\{\mathrm{coef}_f(m) : m \in S\}$ *replaced by* $\mathrm{span}_\mathbb{K}(f)$ *and* $\mathrm{span}_\mathbb{K}\{\mathrm{coef}_f(m) : m \in S\}$ *respectively.*

We use low-degree concentration of f only for univariate polynomials f in this paper.

Hitting-Sets. We say a set $\mathcal{H} \subseteq \mathbb{F}^n$ is a *hitting-set* for a nonzero polynomial $f \in \mathbb{F}[x_1, \ldots, x_n]$ if there exists $\alpha \in \mathcal{H}$ such that $f(\alpha) \neq 0$. We say $\mathcal{H} \subseteq \mathbb{F}^n$ is a hitting-set for a class of polynomials $\mathcal{C} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ if $\mathcal{H}$ is a hitting-set for every nonzero polynomial in $\mathcal{C}$.

ROABPs. A *read-once oblivious arithmetic branching program* (ROABP) in the order $x_1, \ldots, x_n$ is a weighted directed graph $\mathcal{B}$ with $n+1$ layers of vertices $\{V_0, V_1, \ldots, V_n\}$ together with a start node s and an end node t. All the edges are from s to V_0, V_{i-1} to V_i for $i \in [n]$, or V_n to t.

For $i \in [n]$, the weight of an edge e from V_{i-1} to V_i is a univariate polynomial $w_e \in \mathbb{F}[x_i] \subseteq \mathbb{F}[\mathbf{x}]$. The weights of the edges e from s to V_0 and those from V_n to t are constants (i.e, $w_e \in \mathbb{F}$). We define the weight of a path in $\mathcal{B}$ from s to t to be the product of the weights of the edges on that path. The polynomial computed by $\mathcal{B}$ is the sum of the weights of the paths in $\mathcal{B}$ from s to t.

Let $r = \max\{|V_i| : i \in [n]\}$. We say $\mathcal{B}$ has *length*[4] n and *width* r. We say $\mathcal{B}$ has *individual degree* d if $\deg(w_e) \leq d$ for $e \in E(\mathcal{B})$. By adding dummy vertices, we may always assume each layer V_i of $\mathcal{B}$ has exactly r vertices. The polynomial f computed by $\mathcal{B}$ can be represented as a product of matrices $f = \beta^\top f_1 f_2 \cdots f_n \gamma$ where $\beta, \gamma \in \mathbb{F}^{r \times 1}$ and $f_i \in \mathbb{A}[x_i]$ for $i \in [n]$ with $\mathbb{A} = M_{r \times r}(\mathbb{F})$.

Let $c > 0$ be a large enough constant. Throughout the paper, we always assume the length n of an ROABP is at least c and the width r is at least two, which is fine since explicit hitting-sets of polynomial size for ROBAPs are easy to construct when $n < c$ or $r = 1$. These assumptions are made to avoid technicalities in boundary cases (e.g. $\log \log r$ is undefined when $r = 1$). Similarly, we always assume the individual degree bound d is at least c by replacing d with $\max\{d, c\}$ if necessary.

Unknown-Order and Any-Order ROABPs. The above definition of ROABPs is given with respect to the variable order $x_1, x_2, \ldots, x_n$. More generally, we say an ROABP has an *unknown order* or is an *unknown-order ROABP* if it has the variable order $x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$ where π is an arbitrary permutation of $[n]$.

Let $\mathcal{C}$ be a class of unknown-order ROABPs. We say $f \in \mathbb{F}[\mathbf{x}]$ is computed by ROABPs in $\mathcal{C}$ *in any order* (or simply an *any-order ROABP* in $\mathcal{C}$) if for every permutation π of $[n]$, f is computed by an ROABP in $\mathcal{C}$ that has the variable order $x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}$. In this paper, $\mathcal{C}$ will be the class of unknown-order ROABPs of length n, width r and individual degree d for some n, r and d.

We also say a polynomial $f \in \mathbb{A}[\mathbf{x}]$ over $\mathbb{A}$ is computed by ROABPs of length n, width r and individual degree d in any order if for any permutation π of $[n]$, we can write $f = f_1 f_2 \cdots f_n$ such that $f_i \in \mathbb{A}[x_{\pi(i)}]$ is a univariate polynomial of degree at most d in $x_{\pi(i)}$ where $\mathbb{A} = M_{r \times r}(\mathbb{F})$.

3 Hitting-Sets for ROABPs

In this section, we give an explicit construction of hitting-sets for ROABPs in an unknown order. Then we prove Theorem 1 and Theorem 2.

[4] The length is also called the *depth* and equals the number of variables.

3.1 Low-Degree Concentration by Random Shift

We start with the following lemma, which states that random shift achieves low-degree concentration with high probability.

▶ **Lemma 5.** *Suppose* $\mathrm{char}(\mathbb{F}) = 0$ *or* $\mathrm{char}(\mathbb{F}) > d \in \mathbb{N}$. *Let* $f \in \mathbb{A}[x]$ *be a univariate polynomial of degree* d. *Let* $\ell = \dim_{\mathbb{F}}(\mathrm{span}(f)) \leq \dim_{\mathbb{F}} \mathbb{A} = r^2$. *Then for all but at most* $d\ell$ *choices of* $\alpha \in \mathbb{F}$, $f(x+\alpha)$ *is* ℓ*-degree concentrated.*

Proof. Using the fact that all the $\ell \times \ell$ minors of the Wronskian matrix $W = \left[\binom{i}{j}\right]_{0 \leq i \leq d, 0 \leq j < \ell}$ are nonzero when $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, it can be shown that $f(x+t)$ is ℓ-degree concentrated over $\mathbb{F}(t)$, where t is an indeterminate. We omit the proof of this claim but note it is a special case of [7, Theorem 2] applied to univariate polynomials.

View $\mathbb{A}$ as a vector space over $\mathbb{F}$. As $f(x+t)$ is ℓ-degree concentrated, the matrix formed by the ℓ vectors $\mathrm{coef}_{f(x+t)}(1), \mathrm{coef}_{f(x+t)}(x), \ldots, \mathrm{coef}_{f(x+t)}(x^{\ell-1}) \in \mathbb{A}$ has a nonzero $\ell \times \ell$ minor $g(t)$. Note $g(t)$ is a polynomial in t of degree at most $d\ell$. So for all but at most $d\ell$ choices of α, we have $g(\alpha) \neq 0$. For such α, the vectors $\mathrm{coef}_{f(x+\alpha)}(1), \mathrm{coef}_{f(x+\alpha)}(x), \ldots, \mathrm{coef}_{f(x+\alpha)}(x^{\ell-1})$ are linearly independent and hence span the space $\mathrm{span}(f) = \mathrm{span}(f(x+\alpha))$. So $f(x+\alpha)$ is ℓ-degree concentrated. ◀

We use Lemma 5 to preprocess the univariate polynomials f_i in an ROABP so that they are r^2-degree concentrated: Suppose $f_1 \in \mathbb{A}[x_1], f_2 \in \mathbb{A}[x_2] \ldots, f_n \in \mathbb{A}[x_n]$ are univariate polynomials of degree at most d. Let $S \subseteq \mathbb{F}$ such that $|S| > ndr^2$. By Lemma 5 and the union bound, there exists $\alpha \in S$ such that $f_i(x_i + a)$ is r^2-degree concentrated for $i \in [n]$.

▶ Remark. Lemma 5 may not hold if $0 < \mathrm{char}(\mathbb{F}) \leq d$. For example, let $a, b \in \mathbb{A}$ be linearly independent over a field $\mathbb{F}$ of characteristic $p > 0$. Let $f(x) = ax^d + b$ where $d \geq p$ is a power of p. Then $f(x+\alpha) = ax^d + \alpha^d a + b$ is not ℓ-degree concentrated for $\alpha \in \mathbb{F}$ and $\ell \leq d$.

3.2 Basis Isolation

A *weight assignment* of the variables $x_1, x_2, \ldots, x_n$ is a map $w : \{x_1, x_2, \ldots, x_n\} \to \mathbb{N}$. Extend w to a map $w : \mathcal{M}(\mathbf{x}) \to \mathbb{N}$ on the set $\mathcal{M}(\mathbf{x})$ of monomials by $w(\mathbf{x}^{\mathbf{a}}) := \sum_{i=1}^{n} w(x_i)a_i$ for $\mathbf{x}^{\mathbf{a}} = \prod_{i=1}^{n} x_i^{a_i} \in \mathcal{M}(\mathbf{x})$.

One basic tool we need is the following explicit construction of weight assignments that separate polynomially many monomials.

▶ **Lemma 6** ([1, Lemma 4, restated])**.** *For* $n, s, \ell \in \mathbb{N}^+$ *and* $0 < \epsilon < 1$, *there exist weight assignments* $w_1, w_2, \ldots, w_N : \{x_1, x_2, \ldots, x_n\} \to [N \log N]$, *where* $N = \mathrm{poly}(n, s, \log \ell, \epsilon^{-1})$, *such that for any* s *monomials* $m_1, m_2, \ldots, m_s \in \mathcal{M}(\mathbf{x})$ *of individual degree less than* ℓ, *all but at most* ϵ*-fraction of* w_i *among* $w_1, w_2, \ldots, w_N$ *separate these monomials, i.e.,* $w_i(m_j) \neq w_i(m_{j'})$ *for* $j, j' \in [s]$ *with* $m_j \neq m_{j'}$. *The weight assignments* $w_1, w_2, \ldots, w_N$ *can be computed in time polynomial in* N.

We are interested in weight assignments that have the property of *basis isolation*, introduced in [1].

▶ **Definition 7** (basis isolating weight assignment [1])**.** *For a polynomial* $f \in \mathbb{A}[\mathbf{x}]$, *we say* $w : \{x_1, x_2, \ldots, x_n\} \to \mathbb{N}$ *is a* basis isolating weight assignment *for* f *if there exists a set* $S \subseteq \mathcal{M}(\mathbf{x})$ *of monomials whose coefficients in* f *form a basis of* $\mathrm{span}(f)$, *such that*

1. $w(m) \neq w(m')$ *for distinct* $m, m' \in S$, *and*
2. $\mathrm{coef}_f(m) \in \mathrm{span}\{\mathrm{coef}_f(m') : m' \in S,\ w(m') < w(m)\}$ *for* $m \in \mathcal{M}(\mathbf{x}) \setminus S$.

The following lemma states that, if w is a basis isolating weight assignment, then the variable substitution map $x_i \mapsto y^{w(x_i)}$ preserves the nonzeroness of polynomials. This makes basis isolating weight assignments a very useful tool for PIT.

▶ **Lemma 8** ([1, Lemma 6]). *Let $f(\mathbf{x}) \in \mathbb{A}[\mathbf{x}]$, $\beta, \gamma \in \mathbb{F}^r$, and $g(\mathbf{x}) = \beta^\top f(\mathbf{x})\gamma \in \mathbb{F}[\mathbf{x}]$. Suppose $w : \{x_1, x_2, \ldots, x_n\} \to \mathbb{N}$ is a basis isolating weight assignment for $f \in \mathbb{A}[\mathbf{x}]$. Then $g(\mathbf{x}) = 0$ iff $g(y^{w(x_1)}, y^{w(x_2)}, \ldots, y^{w(x_n)}) = 0$.*

Explicit Construction. We use the following explicit construction of basis isolating weight assignments for ROABPs, which is a k-ary generalization of the one in [1].

Let $n, \ell \in \mathbb{N}^+$, $k \in \{2, \ldots n\}$ and $\epsilon \in (0, 1)$ where n is a power of k. Let $u = \log n / \log k \in \mathbb{N}$. Let $N = \text{poly}(n, s, \log \ell, \epsilon^{-1})$ and $w_1, w_2, \ldots, w_N$ be as in Lemma 6 with respect to the parameters n, s, ℓ, ϵ, where $s = \max\{\ell, r^{2k}\}$. Let $h = n\ell N \log N$. For $\mathbf{t} = (t_1, t_2, \ldots, t_u) \in [N]^u$, define the weight assignment $w_{\mathbf{t}} : \{x_1, x_2, \ldots, x_n\} \to [N \log N h^u]$ by

$$w_{\mathbf{t}}(x_i) = \sum_{j=1}^{u} w_{t_j}(x_i) h^{u-j}.$$

So $w_{\mathbf{t}}$ is a linear combination of $w_{t_1}, w_{t_2}, \ldots, w_{t_u}$, where w_{t_j} is multiplied by h^{u-j} for $j \in [u]$. If $u = 0$ (i.e., $n = 1$), define $w_{\mathbf{t}}(x_1) = 1$ instead for the unique element $\mathbf{t} \in [N]^u$.

▶ **Lemma 9.** *Let $\pi : [n] \to [n]$ be a permutation. Let $f = \prod_{i=1}^{n} f_i$ where $f_i \in \mathbb{A}[x_{\pi(i)}]$ is ℓ-degree concentrated for $i \in [n]$. Then for all but at most ϵ'-fraction of $\mathbf{t} \in [N]^u$, $w_{\mathbf{t}}$ is a basis isolating weight assignment for f, where $\epsilon' = \frac{(n-1)\epsilon}{k-1}$.*

Proof. We prove the lemma for the case $(x_{\pi(1)}, x_{\pi(2)}, \ldots, x_{\pi(n)}) = (x_1, x_2, \ldots, x_n)$. The same proof works for arbitrary variable orders since the monomial separation property of w_i as asserted in Lemma 6 is closed under any permutation of the variables $x_1, x_2, \ldots, x_n$.

The proof is done by induction on $u = \log n / \log k$.

Base case: Suppose $u = 0$, i.e., $n = 1$. We want to prove that the weight assignment $w_{\mathbf{t}}$ defined by $w_{\mathbf{t}}(x_1) = 1$ is basis isolating for $f = f_1$. Choose the set of monomials $S \subseteq \mathbb{F}[x_1]$ in the following greedy way: Start with $S = \emptyset$ and enumerate $i = 0, 1, 2 \ldots$. For each i, add x^i to S whenever $\text{coef}_{f_1}(x_i) \notin \text{span}\{\text{coef}_{f_1}(m) : m \in S\}$. Continue this process until $\text{span}\{\text{coef}_{f_1}(m) : m \in S\} = \text{span}(f)$. Then $w_{\mathbf{t}}$ and S satisfy Definition 7. So $w_{\mathbf{t}}$ is a basis isolating weight assignment for $f = f_1$.

Inductive step: Suppose $u > 0$, and assume the claim holds for $u' = u-1$. Let $n' = n/k = k^{u'}$. Divide $[n]$ into k blocks $B_1, B_2, \ldots, B_k$, where $B_i = \{(i-1)n'+1, (i-1)n'+2, \ldots, in'\}$. For $i \in [k]$, let $f^{(i)} = \prod_{j \in B_i} f_j = \prod_{j=(i-1)n'+1}^{in'} f_j$. So $f = \prod_{i=1}^{k} f^{(i)}$.

Let $\epsilon'' = \frac{(n'-1)\epsilon}{k-1}$. By the induction hypothesis and the union bound, for all but at most $k\epsilon''$-fraction of $\mathbf{t} = (t_1, t_2, \ldots, t_{u-1}) \in [N]^{u-1}$, $w_{\mathbf{t}}$ is a basis isolating weight assignment for $f^{(1)}, f^{(2)}, \ldots, f^{(k)}$. Fix such $\mathbf{t}$. For $i \in [k]$, let $S_i \subseteq \mathcal{M}(x_j : j \in B_i)$ be a set of monomials in the variables x_j with $j \in B_i$ such that $w_{\mathbf{t}}$, S_i and $f^{(i)}$ satisfy the conditions in Definition 7. Then $|S_i| \leq \dim_{\mathbb{F}} \text{span}(f^{(i)}) \leq \dim_{\mathbb{F}} \mathbb{A} = r^2$ for $i \in [k]$.

▷ Claim 10. The monomials in $S_1, S_2, \ldots, S_k$ have individual degree less than ℓ.

Proof of Claim 10. Assume to the contrary that some S_i contains a monomial m whose degree in some variable x_j is $d \geq \ell$. Write $m = x_j^d \bar{m}$ where $\bar{m}$ does not depend on x_j. As f_j is ℓ-degree concentrated, we have $\text{coef}_{f_j}(x_j^d) \in \text{span}\{\text{coef}_{f_j}(x_j^a) : 0 \leq a < \ell\}$. Using the fact $f^{(i)} = \prod_{j \in B_i} f_j$ and $f_j \in \mathbb{A}[x_j]$ for $j \in S_i$, we see

$$\text{coef}_{f^{(i)}}(m) \in \text{span}\{\text{coef}_{f^{(i)}}(x_j^a \bar{m}) : 0 \leq a < \ell\} \subseteq \text{span}\{\text{coef}_{f^{(i)}}(m') : w_{\mathbf{t}}(m') < w_{\mathbf{t}}(m)\}.$$

But, from (2) of Definition 7, we know that for any $m' \in \mathcal{M}(x_j : j \in B_i) \setminus S_i$,

$$\text{coef}_{f^{(i)}}(m') \in \text{span}\{\text{coef}_{f^{(i)}}(m'') : m'' \in S_i,\ w_{\mathbf{t}}(m'') < w_{\mathbf{t}}(m')\}.$$

From the above two containments we get that

$$\text{coef}_{f^{(i)}}(m) \in \text{span}\{\text{coef}_{f^{(i)}}(m'') : m'' \in S_i,\ w_{\mathbf{t}}(m'') < w_{\mathbf{t}}(m)\}.$$

This contradicts the fact that the coefficients of the monomials in S_i form a basis of $\text{span}(f_i)$ (Definition 7). ◁

Let $T := \{\prod_{i=1}^{k} m_i : m_i \in S_i \text{ for } i \in [k]\}$. Then $\text{span}\{\text{coef}_f(m) : m \in T\} = \text{span}(f)$. Note $|T| = \prod_{i=1}^{k} |S_i| \le r^{2k} \le s$, and T consists of monomials of individual degree less than ℓ. By Lemma 6, for all but at most ϵ-fraction of $t_u \in [N]$, w_{t_u} separates the monomials in T. Fix such t_u and let $\mathbf{t}' = (t_1, t_2, \ldots, t_u) = (\mathbf{t}, t_u)$.

For $m \in T$, as the individual degree of m is less than ℓ, we have $0 \le w_{t_u}(m) < n\ell N \log N = h$. By definition, $w_{\mathbf{t}'}(m) = w_{t_u}(m)$ if $u = 1$ and $w_{\mathbf{t}'}(m) = w_{\mathbf{t}}(m)h + w_{t_u}(m)$ if $u > 1$. In either case, we have $w_{\mathbf{t}'}(m) \neq w_{\mathbf{t}'}(m')$ whenever $w_{t_u}(m) \neq w_{t_u}(m')$ for $m, m' \in T$. Therefore, the weight assignment $w_{\mathbf{t}'}$ also separates the monomials in T.

Next, we choose a subset $S \subseteq T$ of monomials such that $w_{\mathbf{t}'}$, S, and f satisfy the conditions in Definition 7. Initially, let $S = \emptyset$. Choose $m \in T$ with the minimum weight $w_{\mathbf{t}'}(m)$ such that $\text{coef}_f(m) \notin \text{span}\{\text{coef}_f(m') : m' \in S\}$, and add m to S. Note m is unique as $w_{\mathbf{t}'}$ separates the monomials in T. Repeat this step until $\text{span}\{\text{coef}_f(m) : m \in S\}$ equals $\text{span}\{\text{coef}_f(m) : m \in T\} = \text{span}(f)$.

We check that $w_{\mathbf{t}'}$, S and f satisfy the conditions in Definition 7. The set $\{\text{coef}_f(m) : m \in S\}$ is a basis of $\text{span}(f)$ by our choice of S. As $w_{\mathbf{t}'}$ separates the monomials in $T \supseteq S$, Condition (1) of Definition 7 holds. We now prove that Condition (2) also holds.

▷ Claim 11. $\text{coef}_f(m) \in \text{span}\{\text{coef}_f(m') : m' \in S,\ w_{\mathbf{t}'}(m') < w_{\mathbf{t}'}(m)\}$ for $m \in \mathcal{M}(\mathbf{x}) \setminus S$.

Proof of Claim 11. We prove the claim by induction on $w := w_{\mathbf{t}'}(m)$. The claim is vacuously true for $w < 0$ (since this is impossible). Now suppose $w \ge 0$ and the claim holds for $w' < w$. If $m \in T$, the claim holds by our choice of S. So assume $m \notin T$. Write $m = \prod_{i=1}^{k} m_i$ where m_i is a monomial in the variables in B_i. As $m \notin T$, there exists $i \in [k]$ such that $m_i \notin S_i$. Assume $i = 1$ (the other cases are similar). By the choice of S_1, we have

$$\begin{aligned}\text{coef}_{f^{(1)}}(m_1) &\in \text{span}\{\text{coef}_{f^{(1)}}(m_1') : m_1' \in S_1,\ w_{\mathbf{t}}(m_1') < w_{\mathbf{t}}(m_1)\} \\ &\subseteq \text{span}\{\text{coef}_{f^{(1)}}(m_1') : m_1' \in S_1,\ w_{\mathbf{t}'}(m_1') < w_{\mathbf{t}'}(m_1)\}.\end{aligned}$$

where the second step holds since $w_{t_u}(m_1') < n\ell N \log N = h$ for $m_1' \in S_1$, which in turn holds by Claim 10. Therefore

$$\begin{aligned}\text{coef}_f(m) &\in \text{span}\{\text{coef}_f(m_1' m_2 \cdots m_k) : m_1' \in S_1,\ w_{\mathbf{t}'}(m_1') < w_{\mathbf{t}'}(m_1)\} \\ &\subseteq \text{span}\{\text{coef}_f(m') : m' \in \mathcal{M}(\mathbf{x}), w_{\mathbf{t}'}(m') < w_{\mathbf{t}'}(m)\}.\end{aligned}$$

Consider a monomial $m' \in \mathcal{M}(\mathbf{x})$ satisfying $w_{\mathbf{t}'}(m') < w_{\mathbf{t}'}(m)$. By the induction hypothesis, either $m' \in S$, or $\text{coef}_f(m')$ is in the span of the coefficients of those monomials in S with weight strictly less than $w_{\mathbf{t}'}(m') < w_{\mathbf{t}'}(m)$. It follows that $\text{coef}_f(m) \in \text{span}\{\text{coef}_f(m') : m' \in S, w_{\mathbf{t}'}(m') < w_{\mathbf{t}'}(m)\}$. ◁

By the union bound, for all but at most ϵ'-fraction of $\mathbf{t}' = (t_1, \ldots, t_u) \in [N]^u$, where $\epsilon' = k\epsilon'' + \epsilon = \frac{(n-1)\epsilon}{k-1}$, $w_{\mathbf{t}'}$ is a basis isolating weight assignment for f. This completes the proof for the inductive step. ◀

Let $\epsilon = 1/n$. Then the maximum values of the weight assignments $w_{\mathbf{t}}$ constructed above are polynomial in h^u with $h = \mathrm{poly}(n, \ell, r^k)$ and $u = \log n / \log k$, which suggests that we should choose $k = \Theta(\log(n\ell)/\log r)$. However, as $k \in \{2, \ldots, n\}$, we have to choose $k = 2$ (resp. $k = n$) when $\log(n\ell)/\log r$ is subconstant (resp. superlinear in n). This yields the following theorem.

▶ **Theorem 12.** *Let $\mathcal{C}$ be the family of polynomials $f = \beta^\top f_1 f_2 \cdots f_n \gamma$ computed by ROABPs of length n, width r and individual degree d in an unknown order, where each f_i is ℓ-degree concentrated. Then there exists an explicit hitting-set for $\mathcal{C}$ of size polynomial in $M_0(n, r, d, \ell)$, where*

$$M_0(n,r,d,\ell) := \begin{cases} (nr)^{\log n} d & n\ell \le r^2, \\ (n\ell)^{\frac{\log n}{\log\log(n\ell) - \log\log r}} d & r^2 < n\ell < r^n, \\ n\ell d & n\ell \ge r^n. \end{cases}$$

Proof. Choose

$$k = \begin{cases} 2 & n\ell \le r^2, \\ \log(n\ell)/\log r & r^2 < n\ell < r^n, \\ n & n\ell \ge r^n. \end{cases}$$

By adding dummy variables, we may assume n is a power of k. Let $\epsilon = 1/n$. Construct the weight assignment $w_{\mathbf{t}} : \{x_1, x_2, \ldots, x_n\} \to [N \log N h^u]$ for $\mathbf{t} \in [N]^u$ as above, where $N = \mathrm{poly}(n, s, \log \ell, \epsilon^{-1})$, $s = \max\{\ell, r^{2k}\}$, $h = n\ell N \log N$, and $u = \log n / \log k$.

Consider $0 \neq f = \beta^\top f_1 f_2 \cdots f_n \gamma \in \mathcal{C}$. By Lemma 9, there exists $\mathbf{t} \in [N]^u$ such that $w_{\mathbf{t}}$ is a basis isolating weight assignment for $f_1 f_2 \cdots f_n \in \mathbb{A}[\mathbf{x}]$, which implies that $g_{w_{\mathbf{t}}}(y) := f(y^{w_{\mathbf{t}}(x_1)}, \ldots, y^{w_{\mathbf{t}}(x_n)}) \neq 0$ by Lemma 8. As $g_{w_{\mathbf{t}}}(y)$ is univariate, any subset of $\mathbb{F}$ of size $\deg(g_{w_{\mathbf{t}}}) + 1 \le N \log N h^u n d + 1$ is a hitting-set for $g_{w_{\mathbf{t}}}$. Enumerating all $\mathbf{t} \in [N]^u$, we obtain an explicit hitting-set for $\mathcal{C}$ of size at most $N^u(N \log N h^u n d + 1)$, which is polynomial in $M_0(n, r, d, \ell)$. ◀

Theorem 1 follows easily from Theorem 12.

Proof of Theorem 1. Note degree-d polynomials are trivially $(d+1)$-degree concentrated. The second part of Theorem 1 (the claim for arbitrary characteristic) then follows from Theorem 12 with $\ell = d + 1$.

Moreover, when $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, we may preprocess the polynomials f_i using Lemma 5 so that they are r^2-degree concentrated. The first part of Theorem 1 then follows from Theorem 12 with $\ell = r^2$. ◀

3.3 Sum of Several ROABPs

Using the reduction in [3], we may extend Theorem 1 to the model of sum of several ROABPs and prove Theorem 2. Here we only sketch the proof as it is the same as the proof in [3] except for some small adjustments.

Proof sketch of Theorem 2. Choose a function $M^*(n, r, d) \ge \mathrm{poly}(n, r, d)$ such that we have explicit hitting-sets of size at most $M^*(n, r, d)$ for the family of polynomials computed by ROABPs of length n, width r and individual degree d in an unknown order. By Theorem 1, we may choose $M^*(n, r, d)$ to be polynomial in $M(n, r, d)$ when $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$ and polynomial in $M'(n, r, d)$ in arbitrary characteristic.

Fix n and d. It was shown in the proof of [3, Lemma 14] (and the proof of [3, Lemma 12]) that for $c \geq 1$ and $r \geq 2$, one can explicitly construct a set $\mathcal{S}$ of ring homomorphisms $\Psi : \mathbb{F}[x_1, x_2, \ldots, x_n] \to \mathbb{F}[t]$ satisfying the following properties ($\mathcal{C}$ is the class of sum of c ROABPs):

- If $0 \neq f \in \mathcal{C}$, there exists $\Psi \in \mathcal{S}$ such that $\Psi(f) \neq 0$.
- For $f \in \mathcal{C}$ and $\Psi \in \mathcal{S}$, the degree of $\Psi(f)$ is at most $S(c, r)$, where $S(1, r) \leq M^*(n, r, d)$ and

$$S(c, r) \leq \text{poly}(M^*(n, r, d)) \cdot S(c - 1, 2r^3) \tag{1}$$

  for $c \geq 2$.
- The time complexity of computing $\mathcal{S}$ is at most $T(c, r)$, where $T(1, r) \leq \text{poly}(M^*(n, r, d))$ and

$$T(c, r) \leq n2^n \cdot \text{poly}(M^*(n, r, d)) \cdot T(c - 1, 2r^3) \tag{2}$$

  for $c \geq 2$. In particular, the size of $\mathcal{S}$ is bounded by $T(c, r)$.

Solving the recursive relations (1) and (2) above gives $S(c, r) \leq \text{poly}(M^*(n, (2r)^{3^c}, d)^c)$ and $T(c, r) \leq \text{poly}(2^{cn}, M^*(n, (2r)^{3^c}, d)^c)$.

For $f \in \mathcal{C}$ and $\Psi \in \mathcal{S}$, any subset of $\mathbb{F}$ of size $S(c, r)+1$ is a hitting-set for $\Psi(f)$ since $\Psi(f)$ is a univariate polynomial of degree at most $S(c, r)$. Enumerating all possible $\Psi \in \mathcal{S}$, we obtain an explicit hitting-set for $\mathcal{C}$ of size at most $T(c, r)(S(c, r) + 1)$, which is polynomial in $2^{cn} \cdot M(n, (2r)^{3^c}, d)^c$ when $\text{char}(\mathbb{F}) = 0$ or $\text{char}(\mathbb{F}) > d$ and polynomial in $2^{cn} \cdot M'(n, (2r)^{3^c}, d)^c$ in arbitrary characteristic. ◀

4 Hitting-Sets for Any-Order ROABPs

In this section, we prove Theorem 3 by giving an explicit construction of hitting-sets for any-order ROABPs.

4.1 Low-Support Concentration

Following [8, 10], we first achieve low-support concentration by shifting the variables. The basic tool is the following lemma.

▶ **Lemma 13** ([11, Lemma 5.2]). *Suppose $w : \mathbf{x} \to \mathbb{N}$ is a basis isolating weight assignment for $f = \prod_{i=1}^{n} f_i \in \mathbb{A}[\mathbf{x}]$. Then $f(x_1 + y^{w(x_1)}, x_2 + y^{w(x_2)}, \ldots, x_n + y^{w(x_n)})$ is $\lceil \log(r^2 + 1) \rceil$-support concentrated over $\mathbb{F}(y)$.*

The next lemma states that low-support concentration is a "local" property for any-order ROABPs.

▶ **Lemma 14** ([2, 8, 10]). *Let $\ell < n$. Let $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{K}^n$ where $\mathbb{K}$ is an extension field of $\mathbb{F}$. Suppose for any distinct $i_1, i_2 \ldots, i_\ell \in [n]$ and $f_1 \in \mathbb{A}[x_{i_1}], f_2 \in \mathbb{A}[x_{i_2}], \ldots, f_\ell \in \mathbb{A}[x_{i_\ell}]$ of degree at most d, the product $f_1(x_{i_1} + s_{i_1}) f_2(x_{i_2} + s_{i_2}) \cdots f_\ell(x_{i_\ell} + s_{i_\ell})$ is ℓ-support concentrated over $\mathbb{K}$. Then for $f(\mathbf{x}) \in \mathbb{A}[\mathbf{x}]$ computed by ROABPs of length n, width r and individual degree d in any order, $f(\mathbf{x} + \mathbf{s})$ is ℓ-support concentrated over $\mathbb{K}$.*

Explicit Construction. Let $\mathbb{K} = \mathbb{F}(y, z, t)$, where y, z and t are indeterminates. We construct $\mathbf{s} \in \mathbb{F}[y, z, t]^n$ such that the shift $\mathbf{x} \mapsto \mathbf{x} + \mathbf{s}$ achieves low-support concentration over $\mathbb{K}$ for polynomials computed by any-order ROABPs. The construction is as follows.

- Let $\ell = r^2$ if $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$. Otherwise let $\ell = d + 1$.
- Choose sufficiently large $\bar{n} = \mathrm{poly}(\log r)$ and let $\mathcal{H} = \{h : [n] \to [\bar{n}]\}$ be an explicit family of hash functions of size $\mathrm{poly}(n, \log r)$ such that for any $T \subseteq [n]$ of size $\lceil \log(r^2+1) \rceil$, there exists $h \in \mathcal{H}$ that maps T injectively to $[\bar{n}]$. Such an explicit family $\mathcal{H}$ can be constructed using pairwise independence [5].
- Using Lemma 9, construct a set $\mathcal{S}$ of weight assignments $w : \{u_1, u_2, \ldots, u_{\bar{n}}\} \to \mathbb{N}$ of the variables $u_1, u_2, \ldots, u_{\bar{n}}$ such that for any permutation $\pi : [\bar{n}] \to [\bar{n}]$ and polynomials $f_1 \in \mathbb{A}[u_1], f_2 \in \mathbb{A}[u_2], \ldots, f_{\bar{n}} \in \mathbb{A}[u_{\bar{n}}]$ of degree at most d that are ℓ-degree concentrated, there exists a basis isolating weight assignment in $\mathcal{S}$ for $\prod_{i=1}^{\bar{n}} f_{\pi(i)}$.
- Fix an injective map $\psi : \mathcal{S} \times \mathcal{H} \to \mathbb{F}$. For $(w, h) \in \mathcal{S} \times \mathcal{H}$, construct the polynomial $p_{w,h}(z) \in \mathbb{F}[z]$ of degree $|\mathcal{S} \times \mathcal{H}| - 1$ by interpolation such that for $(w', h') \in \mathcal{S} \times \mathcal{H}$,

$$p_{w,h}(\psi(w', h')) = \begin{cases} 1 & (w', h') = (w, h) \\ 0 & (w', h') \neq (w, h). \end{cases}$$

- Define $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{F}[y, z, t]^n$ by $s_i(y, z, t) = t + \sum_{(w,h) \in \mathcal{S} \times \mathcal{H}} y^{w(u_{h(i)})} p_{w,h}(z)$ for $i \in [n]$. So $s_i(y, \psi(w, h), t) = t + y^{w(u_{h(i)})}$ for $i \in [n]$ and $(w, h) \in \mathcal{S} \times \mathcal{H}$.

The main difference between the above construction and the one in [10] is the use of hash functions, which has the effect of reducing the number of variables from n to $\bar{n} = \mathrm{poly}(\log r)$ in the analysis.

▶ **Lemma 15.** *Suppose $f \in \mathbb{A}[\mathbf{x}]$ is computed by ROABPs of length n, width r and individual degree d in any order. Then $f(\mathbf{x} + \mathbf{s})$ is $\lceil \log(r^2+1) \rceil$-support concentrated over $\mathbb{K} = \mathbb{F}(y, z, t)$.*

Proof. The lemma is trivial if $n < \lceil \log(r^2 + 1) \rceil$. So assume $n \geq \lceil \log(r^2 + 1) \rceil$. Let $n' = \lceil \log(r^2+1) \rceil$. Consider distinct $i_1, i_2 \ldots, i_{n'} \in [n]$ and $f_1 \in \mathbb{A}[x_{i_1}], f_2 \in \mathbb{A}[x_{i_2}], \ldots, f_{n'} \in \mathbb{A}[x_{i_{n'}}]$ of degree at most d. By Lemma 14, it suffices to prove that

$$g := \prod_{j=1}^{n'} f_j(x_{i_j} + s_{i_j}) \in (\mathbb{A}[y, z, t])[\mathbf{x}] \subseteq (\mathbb{A} \otimes_{\mathbb{F}} \mathbb{K})[\mathbf{x}]$$

is n'-support concentrated over $\mathbb{K}$.

Fix $h \in \mathcal{H}$ such that h maps $\{i_1, \ldots, i_{n'}\}$ injectively to $[\bar{n}]$. Note that there exists $\alpha \in \mathbb{F}$ such that for $j \in [n']$, $f_j(x_{i_j} + \alpha)$ is ℓ-degree concentrated: If $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, then $\ell = r^2$ and this claim follows from Lemma 5. Otherwise, $\ell = d + 1$ and this claim holds trivially. Fix such α.

Let $f^*(u_1, u_2, \ldots, u_{\bar{n}}) := \prod_{j=1}^{n'} f_j(u_{h(i_j)} + \alpha) \in \mathbb{A}[u_1, u_2, \ldots, u_{\bar{n}}]$. By the choice of $\mathcal{S}$, there exists a basis isolating weight assignment $w : \{u_1, u_2, \ldots, u_{\bar{n}}\} \to \mathbb{N}$ in $\mathcal{S}$ for f^*. Fix such w. By Lemma 13, $f^*(u_1 + y^{w(u_1)}, \ldots, u_{\bar{n}} + y^{w(u_{\bar{n}})}) = \prod_{j=1}^{n'} f_j(u_{h(i_j)} + y^{w(u_{h(i_j)})} + \alpha)$ is n'-support concentrated over $\mathbb{F}(y)$. Substituting $u_{h(i_j)}$ with x_{i_j} for $j \in [n']$, we see that

$$g^* := \prod_{j=1}^{n'} f_j(x_{i_j} + y^{w(u_{h(i_j)})} + \alpha) \in (\mathbb{A}[y])[\mathbf{x}] \subseteq (\mathbb{A} \otimes_{\mathbb{F}} \mathbb{F}(y))[\mathbf{x}]$$

is n'-support concentrated over $\mathbb{F}(y)$.

Let $g_0 = \prod_{j=1}^{n'} f_j(x_{i_j})$. As $g_0(\mathbf{x} + \alpha) = g^*|_{y=0}$, we have $\mathrm{span}_{\mathbb{F}(y)}(g_0) = \mathrm{span}_{\mathbb{F}(y)}(g_0(\mathbf{x} + \alpha)) \subseteq \mathrm{span}_{\mathbb{F}(y)}(g^*)$. On the other hand, note the coefficients of g^* can be written as linear combinations of those of g_0 over $\mathbb{F}(y)$. So $\mathrm{span}_{\mathbb{F}(y)}(g^*) \subseteq \mathrm{span}_{\mathbb{F}(y)}(g_0)$. It follows that $\mathrm{span}_{\mathbb{F}(y)}(g^*) = \mathrm{span}_{\mathbb{F}(y)}(g_0)$. Also note $\mathrm{span}_{\mathbb{K}}(g) = \mathrm{span}_{\mathbb{K}}(g_0)$ since $g(\mathbf{x}) = g_0(\mathbf{x} + \mathbf{s})$.

Let $D := \dim_{\mathbb{F}}(\mathrm{span}(g_0))$. We have $D = \dim_{\mathbb{F}(y)}(\mathrm{span}_{\mathbb{F}(y)}(g_0)) = \dim_{\mathbb{F}(y)}(\mathrm{span}_{\mathbb{F}(y)}(g^*))$ and $D = \dim_{\mathbb{K}}(\mathrm{span}_{\mathbb{K}}(g_0)) = \dim_{\mathbb{K}}(\mathrm{span}_{\mathbb{K}}(g))$. As g^* is n'-support concentrated over $\mathbb{F}(y)$, there exist $m_1, m_2, \ldots, m_D \in \mathcal{M}(\mathbf{x})$ of support size less than n' such that the coefficients $\mathrm{coef}_{g^*}(m_1), \mathrm{coef}_{g^*}(m_2), \ldots, \mathrm{coef}_{g^*}(m_D)$ are linearly independent. Also note $g^* = g|_{z=\psi(w,h),t=\alpha}$, which implies $\mathrm{coef}_{g^*}(m_i) = \mathrm{coef}_g(m_i)|_{z=\psi(w,h),t=\alpha}$ for $i \in [D]$. Therefore, the coefficients $\mathrm{coef}_g(m_1), \mathrm{coef}_g(m_2), \ldots, \mathrm{coef}_g(m_D)$ are also linearly independent. It follows that g is n'-support concentrated over $\mathbb{K}$, as desired. ◀

▶ **Theorem 16.** *Let $\mathbf{s} = (s_1, \ldots, s_n) \in \mathbb{F}[y,z,t]^n$ be as above. Let $T \subseteq \mathbb{F}$ be a large enough set with*

$$|T| = \begin{cases} \mathrm{poly}(n, r^{\log\log r}, d) & \mathrm{char}(\mathbb{F}) = 0 \text{ or } \mathrm{char}(\mathbb{F}) > d \\ \mathrm{poly}(n, r^{\log\log r}, d^{1+\frac{\log\log r}{\max\{1, \log\log d - \log\log r\}}}) & \text{otherwise.} \end{cases}$$

Suppose $f \in \mathbb{A}[\mathbf{x}]$ is computed by ROABPs of length n, width r and individual degree d in any order. Then there exists $(a,b,c) \in T^3$ such that $f(\mathbf{x}+\mathbf{s}(a,b,c)) \in \mathbb{A}[\mathbf{x}]$ is $\lceil \log(r^2+1) \rceil$-support concentrated.

Proof. We know $|\mathcal{H}| = \mathrm{poly}(n, \log r)$. If $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > d$, then $|\mathcal{S}|$ and the maximum value of every $w \in \mathcal{S}$ are bounded by $\mathrm{poly}(M(\bar{n}, r, d)) = \mathrm{poly}(r^{\log\log r}, d)$. Otherwise they are bounded by $\mathrm{poly}(M'(\bar{n}, r, d)) \leq \mathrm{poly}(r^{\log\log r}, d^{1+\frac{\log\log r}{\max\{1, \log\log d - \log\log r\}}})$. The degree of each $s_i \in \mathbb{F}[y,z,t]$ is polynomial in $|\mathcal{S}|$, $|\mathcal{H}|$ and $\max\{w(u_i) : w \in S, i \in [\bar{n}]\}$.

Let $D = \dim_{\mathbb{F}}(\mathrm{span}(f))$. By Lemma 15, $f(\mathbf{x}+\mathbf{s})$ is $\lceil \log(r^2+1) \rceil$-support concentrated over $\mathbb{K}$. So there exist monomials $m_1, m_2 \ldots, m_D \in \mathcal{M}(\mathbf{x})$ of support size less than $\lceil \log(r^2+1) \rceil$ and individual degree at most d such that $\mathrm{coef}_{f(\mathbf{x}+\mathbf{s})}(m_1), \mathrm{coef}_{f(\mathbf{x}+\mathbf{s})}(m_2), \ldots, \mathrm{coef}_{f(\mathbf{x}+\mathbf{s})}(m_D)$ are linearly independen over $\mathbb{K}$. Therefore, the matrix formed by these coefficients (viewed as vectors over $\mathbb{K}$) has a nonzero $D \times D$ minor $g \in \mathbb{K}$. Note g is a polynomial in y, z, t whose degree is polynomial in n, r, d and the degrees of the polynomials $s_i \in \mathbb{F}[y,z,t]$. By the Schwartz-Zippel-DeMillo-Lipton lemma, for large enough T whose size is as in Theorem 16, there exists $(a,b,c) \in T^3$ such that $g(a,b,c) \neq 0$. For such (a,b,c), the coefficients $\mathrm{coef}_{f(\mathbf{x}+\mathbf{s}(a,b,c))}(m_1), \mathrm{coef}_{f(\mathbf{x}+\mathbf{s}(a,b,c))}(m_2), \ldots, \mathrm{coef}_{f(\mathbf{x}+\mathbf{s}(a,b,c))}(m_D)$ span the coefficient span of f, which implies that $f(\mathbf{x}+\mathbf{s}(a,b,c))$ is $\lceil \log(r^2+1) \rceil$-support concentrated. ◀

4.2 Converting Low-Support Concentrated Any-Order ROABPs into Short ROABPs

Our next step follows that in [8, 10], which converts any-order ROABPs with low-support concentration into short ROABPs. In particular, we need the following lemma proved in [8].

▶ **Lemma 17** ([8, Lemma 7.6, restated]). *Let $\mathcal{C}$ be the set of polynomials $f \in \mathbb{F}[\mathbf{x}]$ computed by ROABPs of length n, width r and individual degree d in any order such that f has a monomial m with a nonzero coefficient and $|\mathrm{supp}(m)| < \ell$. Then for some $n' = O(\ell^2)$, there exists an explicit set $S \subseteq (\mathbb{F}[y_1, y_2, \ldots, y_{n'}])^n$ of size $\mathrm{poly}(n,r,d)$ satisfying the following condition: For any $f \in \mathcal{C}$, there exists $(\phi_1, \phi_2, \ldots, \phi_n) \in S$ such that $f(\phi_1, \phi_2, \ldots, \phi_n) \in \mathbb{F}[y_1, y_2, \ldots, y_{n'}]$ is a nonzero polynomial computed by ROABPs of length n', width r and individual degree $n^4(d+1)^2$ in any order.*

Now we are ready to prove Theorem 3.

Proof of Theorem 3. Consider $0 \neq f \in \mathcal{C}$. Regard $f \in \mathbb{F}[\mathbf{x}]$ as an element of $\mathbb{A}[\mathbf{x}] \cong M_{r\times r}(\mathbb{F}[\mathbf{x}])$ such that the $(1,1)$-th entry of the corresponding matrix is f and the other entries are zero. Then f, regarded as an element of $\mathbb{A}[\mathbf{x}]$ this way, is also computed by ROABPs of length n, width r and individual degree d in any order. Let $\mathbf{s} = (s_1, s_2, \ldots, s_n) \in \mathbb{F}[y,z,t]^n$

and $T \subseteq \mathbb{F}$ be as in Theorem 16. By Theorem 16, there exists $(a,b,c) \in T^3$ such that $f^*(\mathbf{x}) := f(\mathbf{x}+\mathbf{s}(a,b,c))$ is $\lceil \log(r^2+1) \rceil$-concentrated. As $f \neq 0$, this implies that $f^*(\mathbf{x})$ has a monomial m with a nonzero coefficient and $|\mathrm{supp}(m)| < \lceil \log(r^2+1) \rceil$.

Let $S \subseteq (\mathbb{F}[y_1, y_2, \ldots, y_{n'}])^n$ be the set in Lemma 17 with $\ell = \lceil \log(r^2+1) \rceil$. By Lemma 17, there exists $\phi = (\phi_1, \phi_2, \ldots, \phi_n) \in S$ such that $f^*(\phi_1, \phi_2, \ldots, \phi_n)$ is a nonzero polynomial computed by ROABPs of length n', width r and individual degree $n^4(d+1)^2$, where $n' = O((\log r)^2)$.

Finally, use Theorem 1 to construct an explicit hitting-set $\mathcal{H}$ for

$$f^*(\phi_1, \phi_2, \ldots, \phi_n) = f(\phi_1 + s_1(a,b,c), \phi_2 + s_2(a,b,c), \ldots, \phi_n + s_n(a,b,c)).$$

Then $\mathcal{H}_{a,b,c,\phi} := \{(\phi_1(\alpha) + s_1(a,b,c), \phi_2(\alpha) + s_2(a,b,c), \ldots, \phi_n(\alpha) + s_n(a,b,c)) : \alpha \in \mathcal{H}\}$ is a hitting-set for f. We do not know the correct $(a,b,c) \in T^3$ and $\phi \in S$ but may just enumerate all the possible choices and then take the union of $\mathcal{H}_{a,b,c,\phi}$. When $\mathrm{char}(\mathbb{F}) = 0$ or $\mathrm{char}(\mathbb{F}) > n^4(d+1)^2$, the size of the final hitting-set we obtain is polynomial in

$$\begin{aligned} |T|^3 \cdot |S| \cdot M(n', r, n^4(d+1)^2) &= \mathrm{poly}(n, r^{\log\log r}, d) \cdot \mathrm{poly}(n, r, d) \cdot M(n', r, n^4(d+1)^2) \\ &= \mathrm{poly}(n, r^{\log\log r}, d). \end{aligned}$$

In arbitrary characteristic, the size of the final hitting-set is polynomial in

$$\begin{aligned} &|T|^3 \cdot |S| \cdot M'(n', r, n^4(d+1)^2) \\ &= \mathrm{poly}(n, r^{\log\log r}, d^{1+\frac{\log\log r}{\max\{1, \log\log d - \log\log r\}}}) \cdot \mathrm{poly}(n, r, d) \cdot M'(n', r, n^4(d+1)^2) \\ &= \mathrm{poly}(r^{\log\log r}, (nd)^{1+\frac{\log\log r}{\max\{1, \log\log(nd) - \log\log r\}}}). \end{aligned}$$

◀

5 Open Problems

We list some open problems.

- The results we have obtained in positive characteristics are worse than those in characteristic zero, due to the issue that random shift may fail to achieve low-degree concentration in positive characteristics. Is it possible to close the gaps between characteristic zero and characteristic $p > 0$?
- In characteristic $p > 0$, are there explicit hitting-sets of polynomial size for any-order ROABPs of length n, width r and individual degree d when $r, d = 2^{O(\log n / \log\log n)}$? The following issue prevents us from obtaining such a result: In Lemma 17, the substitution map $f \mapsto f(\phi_1, \phi_2, \ldots, \phi_n)$ increases the individual degree from d to $n^4(d+1)^2$. Thus an application of Lemma 17 forces the new individual degree to be at least $\mathrm{poly}(n)$ even if we start with $d = n^{o(1)}$.
- It was shown in [7] that in characteristic zero, explicit hitting-sets of size $\mathrm{poly}(s)$ exist for log-variate diagonal circuits of size s. It is a natural question to ask if this result can be extended to commutative or any-order log-variate ROABPs of width $\mathrm{poly}(s)$ and individual degree $\mathrm{poly}(s)$.

References

1 Manindra Agrawal, Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Hitting-sets for ROABP and sum of set-multilinear circuits. *SIAM Journal on Computing*, 44(3):669–697, 2015. doi:10.1137/140975103.

2 Manindra Agrawal, Chandan Saha, and Nitin Saxena. Quasi-polynomial hitting-set for set-depth-Δ formulas. In *Proceedings of the 45th Annual ACM Symposium on Theory of Computing (STOC 2013)*, pages 321–330, 2013. doi:10.1145/2488608.2488649.

3 Pranav Bisht and Nitin Saxena. Poly-time blackbox identity testing for sum of log-variate constant-width ROABPs. *Electronic Colloquium on Computational Complexity (ECCC)*, 2020. URL: https://eccc.weizmann.ac.il/report/2020/042.

4 Mark Braverman, Gil Cohen, and Sumegha Garg. Hitting sets with near-optimal error for read-once branching programs. In *Proceedings of the 50th Annual ACM Symposium on Theory of Computing (STOC 2018)*, page 353–362, 2018. doi:10.1145/3188745.3188780.

5 J. Lawrence Carter and Mark N. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979. doi:10.1016/0022-0000(79)90044-8.

6 Richard A. Demillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Information Processing Letters*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

7 Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP 2018)*, pages 54:1–54:16, 2018. doi:10.4230/LIPIcs.ICALP.2018.54.

8 Michael A. Forbes, Ramprasad Saptharishi, and Amir Shpilka. Hitting sets for multilinear read-once algebraic branching programs, in any order. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing (STOC 2014)*, pages 867–875, 2014. doi:10.1145/2591796.2591816.

9 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs. In *Proceedings of the 54th Annual Symposium on Foundations of Computer Science (FOCS 2013)*, pages 243–252, 2013. doi:10.1109/FOCS.2013.34.

10 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory of Computing*, 13(2):1–21, 2017. doi:10.4086/toc.2017.v013a002.

11 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *Computational Complexity*, 26(4):835–880, 2017. doi:10.1007/s00037-016-0141-z.

12 Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC 1994)*, pages 356–364, 1994. doi:10.1145/195058.195190.

13 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449–461, 1992. doi:10.1007/BF01305237.

14 Øystein Ore. Über höhere Kongruenzen. *Norsk Mat. Forenings Skrifter Ser. I*, 7(15):27, 1922.

15 Ran Raz and Omer Reingold. On recycling the randomness of states in space bounded computation. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC 1999)*, pages 159–168, 1999. doi:10.1145/301250.301294.

16 Ran Raz and Amir Shpilka. Deterministic polynomial identity testing in non-commutative models. *Computational Complexity*, 14(1):1–19, 2005. doi:10.1007/s00037-005-0188-8.

17 Nitin Saxena. Diagonal circuit identity testing and lower bounds. In *Proceedings of the 35th International Colloquium on Automata, Languages, and Programming (ICALP 2008)*, pages 60–71, 2008. doi:10.1007/978-3-540-70575-8_6.

18 Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009.

19 Nitin Saxena. Progress on polynomial identity testing- II. In *Perspectives in Computational Complexity*, volume 26 of *Progress in Computer Science and Applied Logic*, pages 131–146. Springer, 2014. doi:10.1007/978-3-319-05446-9_7.

20 Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

21 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:10.1561/0400000039.

22 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Proceedings of the International Symposiumon on Symbolic and Algebraic Computation*, EUROSAM '79, pages 216–226, 1979. doi:10.1007/3-540-09519-5_73.

Almost Optimal Testers for Concise Representations

Nader H. Bshouty
Department of Computer Science, Technion, Haifa, Israel
bshouty@cs.technion.ac.il

Abstract

We give improved and almost optimal testers for several classes of Boolean functions on n variables that have concise representation in the uniform and distribution-free model. Classes, such as k-Junta, k-Linear, s-Term DNF, s-Term Monotone DNF, r-DNF, Decision List, r-Decision List, size-s Decision Tree, size-s Boolean Formula, size-s Branching Program, s-Sparse Polynomial over the binary field and functions with Fourier Degree at most d.

The approach is new and combines ideas from Diakonikolas et al. [24], Bshouty [13], Goldreich et al. [32], and learning theory. The method can be extended to several other classes of functions over any domain that can be approximated by functions with a small number of relevant variables.

2012 ACM Subject Classification Mathematics of computing; Mathematics of computing → Discrete mathematics; Mathematics of computing → Probabilistic algorithms; Theory of computation → Probabilistic computation

Keywords and phrases Property Testing, Boolean function, Junta

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.5

Category RANDOM

Related Version A full version of the paper is available at `https://arxiv.org/abs/1904.09958`.

Acknowledgements I am extremely grateful to Oded Goldreich for very helpful discussions and comments and for allowing me to include his overview of the first tester in the paper (Section 2). I am also grateful to the reviewers of the earlier version of the manuscript for their helpful comments.

1 Introduction

Property testing of Boolean function was first considered in the seminal works of Blum, Luby and Rubinfeld [12] and Rubinfeld and Sudan [46] and has recently become a very active research area. See for example, [2, 4, 5, 6, 8, 9, 13, 15, 16, 17, 18, 19, 20, 21, 24, 27, 31, 33, 36, 37, 41, 40, 43, 47] and other works referenced in the surveys and books [29, 30, 44, 45].

A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be k-junta if it depends on at most k coordinates. The class k-Junta is the class of all k-juntas. The class k-Junta has been of particular interest to the computational learning theory community [10, 11, 14, 23, 34, 38, 42]. A problem closely related to learning k-Junta is the problem of learning and testing subclasses C of k-Junta and classes C of Boolean functions that can be approximated by k-juntas [9, 11, 25, 17, 24, 32, 33, 43]. In both testing and learning we are given black-box query access to a Boolean function f. In learning, for $f \in C$, we need to learn, with high probability, a hypothesis h that is ϵ-close to f. In testing, for any Boolean function f, we need to distinguish, with high probability, the case that f is in C versus the case that f is ϵ-far from every function in C.

In the *uniform-distribution property testing* (and learning) model, the distance between Boolean functions is measured with respect to the uniform distribution. In the *distribution-free property testing*, [32], (and learning [48]) the distance between Boolean functions is measured with respect to an arbitrary and unknown distribution $\mathcal{D}$ over $\{0,1\}^n$. In the


© Nader H. Bshouty;
licensed under Creative Commons License CC-BY

distribution-free model, the testing (and learning) algorithm is allowed (in addition to making black-box queries) to draw random $x \in \{0,1\}^n$ according to the distribution $\mathcal{D}$. This model is studied in [13, 22, 26, 28, 35, 39].

1.1 Results

In Table 1, we list all the previous results and our results in this paper. In the table, $\tilde{O}(T)$ stands for $O(T \cdot Poly(\log T))$, U and D stand for uniform and distribution-free models, resp., and Exp and Poly stand for exponential and polynomial time, resp.

It follows from the lower bounds of Saglam [47], that our query complexity is almost optimal (up-to log-factor) for the classes k-Junta, k-Linear, k-Term, s-Term DNF, s-Term Monotone DNF, r-DNF (r constant), Decision List, r-Decision List (r constant), size-s Decision Tree, size-s Branching Programs and size-s Boolean Formula.

1.2 Notations

In this subsection, we give some notations that we use throughout the paper.

Denote $[n] = \{1, 2, \ldots, n\}$. For $S \subseteq [n]$ and $x = (x_1, \ldots, x_n)$ we denote $x(S) = \{x_i | i \in S\}$. For $X \subset [n]$ we denote by $\{0,1\}^X$ the set of all binary strings of length $|X|$ with coordinates indexed by $i \in X$. For $x \in \{0,1\}^n$ and $X \subseteq [n]$ we write $x_X \in \{0,1\}^X$ to denote the projection of x over coordinates in X. We denote by 1_X and 0_X the all-one and all-zero strings in $\{0,1\}^X$, respectively. When y is a variable then $z = (y)_X$ is the all y string with coordinates indexed by $i \in X$, i.e., $z_i = y$ for all $i \in X$. When we write $x_I = 0$ we mean $x_I = 0_I$. For $X_1, X_2 \subseteq [n]$ where $X_1 \cap X_2 = \emptyset$ and $x \in \{0,1\}^{X_1}, y \in \{0,1\}^{X_2}$ we write $x \circ y$ to denote their concatenation, i.e., the string in $\{0,1\}^{X_1 \cup X_2}$ that agrees with x over coordinates in X_1 and agrees with y over coordinates in X_2. For $X \subseteq [n]$ we denote $\overline{X} = [n] \backslash X = \{x \in [n] | x \notin X\}$. For a function $f : \{0,1\}^k \to \{0,1\}$, $x \in \{0,1\}^n$ and $X = \{i_1, \ldots, i_k\} \subseteq [n]$ where $i_1 < i_2 < \cdots < i_k$, when we write $f(x_X)$ we mean $f(x_{i_1}, \ldots, x_{i_k})$.

Given $f, g : \{0,1\}^n \to \{0,1\}$ and a probability distribution $\mathcal{D}$ over $\{0,1\}^n$, we say that f is *ϵ-close to g with respect to $\mathcal{D}$* if $\mathbf{Pr}_{x \in \mathcal{D}}[f(x) \neq g(x)] \leq \epsilon$, where $x \in \mathcal{D}$ means x is chosen from $\{0,1\}^n$ according to the distribution $\mathcal{D}$. We say that f is *ϵ-far from g with respect to $\mathcal{D}$* if $\mathbf{Pr}_{x \in \mathcal{D}}[f(x) \neq g(x)] \geq \epsilon$. For a class of Boolean functions C, we say that f is *ϵ-far from every function in C with respect to $\mathcal{D}$* if for every $g \in C$, f is ϵ-far from g with respect to $\mathcal{D}$. We will use U to denote the uniform distribution over $\{0,1\}^n$ or over $\{0,1\}^X$ when X in clear from the context.

For a distribution $\mathcal{D}$ over $\{0,1\}^n$ and $X \subset [n]$, we denote by $\mathcal{D}_X$ the distribution $\mathcal{D}$ projected on the coordinates X. That is, the distribution of x_X when $x \in \mathcal{D}$.

For a Boolean function f and $X \subset [n]$, we say that X is an *influential set* of f if there are $a, b \in \{0,1\}^n$ such that $f(a) \neq f(b_X \circ a_{\overline{X}})$. We call the pair (a, b) (or just a when $b = 0$) a *witness* of f for the influential set X. When $X = \{i\}$ then we say that x_i is an *influential variable* of f and a is *a witness* of f for x_i. Obviously, if X is influential set of f then $x(X)$ contains at least one influential variable of f.

We say that the Boolean function $f : \{0,1\}^n \to \{0,1\}$ is a *literal* (*dictatorship* and *anti-dictatorship*, resp.) if $f \in \{x_1, \ldots, x_n, \overline{x_1}, \ldots, \overline{x_n}\}$ where $\overline{x}$ is the negation of x ($f \in \{x_1, \ldots, x_n\}$ and $f \in \{\overline{x_1}, \ldots, \overline{x_n}\}$, resp.).

Let C be a class of Boolean functions $f : \{0,1\}^n \to \{0,1\}$. We say that C is *symmetric* if for every permutation $\pi : [n] \to [n]$ and every $f \in C$ we have $f_\pi \in C$ where $f_\pi(x) := f(x_{\pi(1)}, \cdots, x_{\pi(n)})$. We say that C is *closed under zero projection* (resp. *closed under one projection*) if for every $f \in C$ and every $i \in [n]$, $f(0_{\{i\}} \circ x_{\overline{\{i\}}}) \in C$ (resp. $f(1_{\{i\}} \circ x_{\overline{\{i\}}}) \in C$). We say it is closed under zero-one projection if is closed under zero and one projection.

Table 1 A table of the results. In the table, $\tilde{O}(T)$ stands for $O(T \cdot poly(\log T))$, U and D stand for uniform and distribution-free models, resp., and Exp and Poly stand for exponential and polynomial time, resp.

Class of Functions	Model	#Queries	Time	Reference
s-Term Monotone DNF	U	$\tilde{O}(s^2/\epsilon)$	Poly.	[43]
s-Term Unate DNF	U	$\tilde{O}(s/\epsilon^2)$	Exp.	[17]
	U	$\tilde{O}(s/\epsilon)$	Poly.	This Paper
s-Term Monotone r-DNF	U	$\tilde{O}(s/\epsilon^2)$	Exp.	[17]
s-Term Unate r-DNF	U	$\tilde{O}(s/\epsilon)$	Poly.	This Paper
	D	$\tilde{O}(s^2r/\epsilon)$	Poly.	This Paper
s-Term DNF	U	$\tilde{O}(s^2/\epsilon)$	Exp.	[24]
	U	$\tilde{O}(s/\epsilon^2)$	Exp.	[17]
	U	$\tilde{O}(s/\epsilon)$	Exp.	This Paper
r-DNF (Constant r)	U	$\tilde{O}(1/\epsilon)$	Poly.	This Paper
Decision List	U	$\tilde{O}(1/\epsilon^2)$	Poly.	[24]
	U	$\tilde{O}(1/\epsilon)$	Poly.	This Paper
Length-k Decision List	D	$\tilde{O}(k^2/\epsilon)$	Poly.	This Paper
r-DL (Constant r)	U	$\tilde{O}(1/\epsilon)$	Poly.	This Paper
k-Linear	U	$\tilde{O}(k/\epsilon)$	Poly.	[7, 12]
	D	$\tilde{O}(k/\epsilon)$	Poly.	This Paper
k-Term	U	$O(1/\epsilon)$	Poly.	[43]
	U	$\tilde{O}(1/\epsilon)$	Poly.	This Paper
	D	$\tilde{O}(k/\epsilon)$	Poly.	This Paper
size-s Decision Trees and	U	$\tilde{O}(s/\epsilon^2)$	Exp.	[17]
size-s Branching Programs	U	$\tilde{O}(s/\epsilon)$	Exp.	This Paper
	D	$\tilde{O}(s^2/\epsilon)$	Exp.	This Paper
size-s Boolean Formulas	U	$\tilde{O}(s/\epsilon^2)$	Exp.	[17]
	U	$\tilde{O}(s/\epsilon)$	Exp.	This Paper
size-s Boolean Circuit	U	$\tilde{O}(s^2/\epsilon^2)$	Exp.	[17]
	U	$\tilde{O}(s^2/\epsilon)$	Exp.	This Paper
Functions with	U	$\tilde{O}(2^{2d}/\epsilon^2)$	Exp.	[17]
Fourier Degree $\leq d$	D	$\tilde{O}(2^d/\epsilon + 2^{2d})$	Poly.	This Paper
s-Sparse Polynomial	U	$poly(s/\epsilon) + \tilde{O}(2^{2d})$	Poly.	[1, 25]
over F_2 of Degree d	U	$\tilde{O}(s^2/\epsilon + 2^{2d})$	Poly.	This Paper+[1]
	U	$\tilde{O}(s/\epsilon + s2^d)$	Poly.	This Paper
	D	$\tilde{O}(s^2/\epsilon + s2^d)$	Poly.	This Paper
s-Sparse Polynomial	U	$\tilde{O}(s/\epsilon^2)$	Exp.	[17]
over F_2	U	$Poly(s/\epsilon)$	Poly.	[25]
	U	$\tilde{O}(s^2/\epsilon)$	Poly.	This Paper

2 Overview of the Distribution-Free Tester

2.1 Preface

Our approach refers to testing properties that are (symmetric) sub-classes C of k-juntas; that is, $f : \{0,1\}^n \to \{0,1\}$ has the property if there exists a function $f' : \{0,1\}^k \to \{0,1\}$ that belongs to a predetermined class C' of functions (over k-bit strings) such that $f(x) = f'(x_\Gamma)$

for some k-subset Γ. Our new approach builds upon the "testing by implicit sampling" approach of Diakonikolas et al. [24], while extending it from the case of uniform distribution to the case of arbitrary unknown distributions (i.e., the distribution-free model).

This allows us to present (almost optimal) *distribution-free* testers for classes of properties that are sub-classes of k-juntas, which correspond to classes of k-bit long Boolean functions.

While we follow Diakonikolas et al. [24] in considering learning algorithms for the underlying classes, our approach is also applicable to testing algorithms (see [30, Sec. 6.2]).

Let us again spell out our task. For a class C of n-bit long Boolean functions and a proximity parameter ϵ, given samples from an unknown distribution $\mathcal{D}$ and oracle access to a function $f : \{0,1\}^n \to \{0,1\}$, we wish to distinguish the case that $f \in C$ from the case that f is ϵ-far from C. Recall that C is a (symmetric) class consisting of a symmetric subclass of k-juntas C'; that is, $f \in C$ if and only if there exists a k-subset $\Gamma \subset [n]$ and $f' \in C'$ such that $f(x) = f'(x_\Gamma)$, where $x_{\{i_1,\ldots,i_k\}} = (x_{i_1}, \ldots, x_{i_k})$. Actually, we also assume that C' is closed under zero projection.

2.2 A Bird's Eye View

The basic strategy is to consider a random partition of $[n]$ to $r = O(k^2)$ parts, denoted $(X_1, \ldots, X_r)$, while relying on the fact that, whp, each X_i contains at most one influential variable. Assuming that $f \in C$, first we determine a set I of at most k indices such that $\cup_{i\in[n]\setminus I} X_i$ contains no "significantly influential" variables of f. Suppose that $f' : \{0,1\}^k \to \{0,1\}$, $f' \in C'$, is a function that corresponds to the tested function $f : \{0,1\}^n \to \{0,1\}$, and that $I \subset [n]$ is indeed the collection of all sets that contain influential variables. The crucial ingredient is devising a method that allows to generate samples of the form $(x', f'(x'))$, when given samples of the form $(x, f(x))$ (for $x \in \mathcal{D}$). We stress that we cannot afford to find the influential variables, and so this method works without determining these locations. Using this method, we can test whether f' belongs to the underlying class C'; hence, we test f by implicitly sampling the projection of $\mathcal{D}$ on the (unknown) influential variables.

The method employed by Diakonikolas et al. [24] only handles the uniform distribution (i.e., the case that $\mathcal{D}$ is uniform over $\{0,1\}^n$), and so it only yields testers for the standard testing model (rather than for the distribution-free testing model). Furthermore, their method as well as the identification of the set I rely heavily on the notion of influence of sets, where the influence of a set S of locations on the value of a function is defined as $\Pr_{x',x''\in\{0,1\}^n : x'_S = x''_S}[f(x') \neq f(x'')]$. However, this notion refers to the uniform distribution (over $\{0,1\}^n$) and does not seem adequate for the distribution-free context (e.g., for[1] $f(x) = x_1 + x_2$ we may get $\Pr_{x',x''\in\mathcal{D} : x'_1 = x''_1}[f(x') \neq f(x'')] = 0$).

We use a different way of identifying the set I and for generating samples for the underlying function f'. Loosely speaking, we identifies I as the set of indices i for which $f(1_{X_i} \circ 0_{\overline{X_i}}) \neq f(0^n)$, where (recall that) $1_S \circ 0_{\overline{S}}$ is a string that is 1 on the locations in S and is 0 on other locations. (*Be warned that this description is an over-simplification*!) This means that for every $i \in I$ and $x \in \{0,1\}^n$, the value of x at the influential variable in the set X_i (a variable whose location is unknown to us!), equals $f(x') + f(0^n)$ where $x' = x_{X_i} \circ 0_{\overline{X}_i}$, i.e., $x'_j = x_j$ if $j \in X_i$ and $x'_j = 0$ otherwise.[2] Note that the foregoing holds when $f \in C$; in

[1] The addition operation in this paper is over the binary field F_2.

[2] Indeed, if $\tau(i) \in X_i$ is the index of the (unique) influential variable that resides in the set X_i, then

$$f(x') = x_{\tau(i)} \cdot f(1_{X_i} \circ 0_{\overline{X_i}}) + (x_{\tau(i)} + 1) \cdot f(0^n) = x_{\tau(i)} + f(0^n)$$

since $f(1_{X_i} \circ 0_{\overline{X_i}}) + f(0^n) = 1$.

general, we can test whether $x \mapsto f(x') + f(0^n)$ is close to a dictatorship (under the uniform distribution) and reject otherwise, whereas if the mapping is close to a dictatorship, we can self-correct it.

To sample the distribution $\mathcal{D}_\Gamma$, where Γ is the influential variables in $X_I = \cup_{i\in I} X_i$, we sample $\mathcal{D}$ and determine the value of the influential variable in each set X_i, for $i \in I$. Queries to the function f' are answered by querying f such that the query $y = y_1 \cdots y_k$ is mapped to the query $\texttt{ext}(y)$ such that[3] $\texttt{ext}(y)_j = y_i$ if j belongs to the i^{th} set in the collection I (and $\texttt{ext}(y)_j = 0$ if $j \in [n] \setminus X_I$). Effectively, we query the function $F : \{0,1\}^n \to \{0,1\}$ defined as $F(x) = f(\texttt{ext}(x_\Gamma))$, and this makes sense provided that F is close to f (under the distribution $\mathcal{D}$). To test the latter hypothesis condition, we sample $\mathcal{D}$ and for each sample point x we compare $f(x)$ to $F(x)$, where here we again use the ability to determine the value of the influential variable in each set. Specifically, $\texttt{ext}(x_\Gamma)$ is computed by determining the value of x_Γ (without knowing Γ), and using our knowledge of $(X_i)_{i\in I}$.

We warn that the foregoing description presumes that we have correctly identified the collection I of all sets containing an influential variable. This leaves us with two questions: The first question is, how do we identify the set I. (Note that the influence of a variable may be as low as 2^{-k}, whereas we seek algorithms of poly(k)-complexity.) The solution (to be presented in Section 2.3.1) will be randomized, and will have one-sided error; specifically, we may fail to identify some sets that contain influential variables, but will never include in our collection sets that have no influential variables. Consequently, $f(1_{X_i} \circ 0_{\overline{X_i}}) \neq f(0^n)$ may not hold for some $i \in I$, and (over-simplifying again) we shall seek instead some $v^{(i)} \in \{0,1\}^n$ such that $f(v^{(i)}) \neq f(w^{(i)})$, where $w^{(i)} = v^{(i)}_{X_i} \circ 0_{X_i}$ (i.e., $w^{(i)}_j = v^{(i)}_j$ if $j \in [n] \setminus X_i$ and $w^{(i)}_j = 0$ otherwise). Second, as before, for every $i \in I$ and $x \in \{0,1\}^n$, we wish to determine the value in x of the influential variable in the set X_i (a variable whose location is unknown to us!). This is done by observing that if $f \in C$ then this value equals $f(x') + f(v^{(i)}) + 1$ where $x' = x_{X_j} \circ v^{(i)}_{\overline{X_j}}$ (i.e., $x'_j = x_j$ if $j \in X_i$ and $x'_j = v^{(i)}_j$ otherwise).[4] Again, we need to test whether $x \mapsto f(x') + f(v^{(i)}) + 1$ is a dictatorship, and use self-correction.

2.3 The Actual Tester

As warned, the above description is an over-simplification, and the actual way in which the set I is identified and used is more complex.

We fix a random partition of $[n]$ to $r = O(k^2)$ parts, denoted $(X_1, \ldots, X_r)$. If $f \in C$, then, with high probability, each X_i contains at most one influential variable, denoted $\tau(i)$. We assume that this is the case when providing intuition throughout this section.

2.3.1 Stage 1: Finding I and corresponding $v^{(i)}$

Our goal is to find a collection I of at most k sets such that the function h_I is $\epsilon/3$-close to f (w.r.t distribution $\mathcal{D}$), where h_I is defined as $h_I(x) = f(x_{X_I} \circ 0_{\overline{X_I}})$ and $X_I = \cup_{i\in I} X_i$. In addition, for each $i \in I$, we seek a witness $v^{(i)}$ for the fact that f depends on some variable in X_i; that is, $f(v^{(i)}) \neq f(w^{(i)})$ for some $v^{(i)}$ that differ from $w^{(i)}$ only on X_i.

---

[3] Notice that $\texttt{ext}(y) = 0_{\overline{X_I}} \circ \left(\underset{i\in I}{\circ} (y_i)_{X_i} \right)$ - Here $(y)_X = 1_X$ if $y = 1$ and 0_X if $y = 0$.

[4] Indeed, if $\tau(i) \in X_i$ is the index of the (unique) influential variable that resides in the set X_i, then

$$f(x') = x_{\tau(i)} \cdot f(v^{(i)}) + (x_{\tau(i)} + 1) \cdot f(w^{(i)}) = x_{\tau(i)} + f(v^{(i)}) + 1$$

since $f(v^{(i)}) + f(w^{(i)}) = 1$.

The procedure

We proceed in iterations, starting with $I = \emptyset$.

1. We sample $\mathcal{D}$ for $O(1/\epsilon)$ times, trying to find $u \in \mathcal{D}$ such that $f(u) \neq h_I(u)$.
   (Note that if $I = \emptyset$, then $h_I(u) = f(0^n)$. In general, we seek u such that $f(u) \neq f(u_{X_I} \circ 0_{\overline{X_I}})$.
   If no such u is found, then we set $h = h_I$ and proceed to Stage 2. In this case, we may assume that h_I is $\epsilon/3$-close to f (w.r.t $\mathcal{D}$).
2. Otherwise (i.e., $f(u) \neq h_I(u)$), we find an $i \in [m] \setminus I$ and $v^{(i)}$ such that $h_I(v^{(i)}) \neq h_{I\cup\{i\}}(v^{(i)})$, which means that X_i contains an influential variable and $v^{(i)}$ is the witness for the sensitivity that we seek. We set $I \leftarrow I \cup \{i\}$ and proceed to the next iteration.
   (We find this i by binary search that seeks i and S such that $h_{I\cup S\cup\{i\}}(u) \neq h_{I\cup S}(u)$, which means that $v^{(i)}$ equals u in locations outside S and is zero on S.)[5]

Once the iterations are suspended (due to not finding u), we reject if $|I| > k$, and continue to the Stage 2 otherwise. Recall that in the latter case $h = h_I$ is $\epsilon/3$-close to f (w.r.t $\mathcal{D}$).

Note that if $f \in C$, then I contains only sets that contain variables of the k-junta, and so we never reject in this stage. In general, if $i \in I$, then $h_{I\setminus\{i\}}(v^{(i)}) \neq h_I(v^{(i)})$, which implies that $f(x') \neq f(x'')$, where x' and x'' differ only on X_i (e.g., $x''_{X_I} = v^{(i)}_{X_I}$ and $x''_j = 0$ if $j \notin X_I$).

2.3.2 Stage 2: Extracting the value of an influential variable

Given a collection I as found in Stage 1 (and a sensitivity witness $v^{(i)}$ for each $i \in I$), let $h = h_I$ and recall that h is close to f w.r.t $\mathcal{D}$. For each $i \in I$, given $x \in \{0,1\}^n$, we wish to determine the value of x at the influential variable that resides in X_i.

For each $i \in I$, we define $\nu_i : \{0,1\}^{|X_i|} \to \{0,1\}$ such that $\nu_i(z) = h_I(y)$, where $y_{X_i} = z$ and $y_{\overline{X_i}} = v^{(i)}_{\overline{X_i}}$. Suppose that $f \in C$, and recall that $\tau(i) \in X_i$ denotes the location of the influential variable in X_i. Let $\sigma(i)$ denote the index of $\tau(i)$ in X_i (i.e., the $\sigma(i)^{\text{th}}$ element of X_i is $\tau(i)$). Then, in this case, ν_i is either a dictatorship or an anti-dictatorship. In particular, if ν_i is a dictatorship, then $\nu_i(z) = z_{\sigma(i)}$ (and otherwise $\nu_i(z) = z_{\sigma(i)} + 1$).

For each $i \in I$, we test whether ν_i is a dictatorship or anti-dictatorship, where testing is w.r.t the uniform distribution over $\{0,1\}^{|X_i|}$. Note that we also check whether ν_i is a dictatorship or anti-dictatorship. If the tester (run with proximity parameter 0.1) fails, we reject. Otherwise (i.e., if we did not reject), we can compute ν_i via self-correction on h_I; that is, to compute ν_i at z, we select $u \in \{0,1\}^{|X_i|}$ at random, and return $\nu_i(z+u) + \nu_i(u)$, which (w.h.p.) equals $(z+u)_{\sigma(i)} + u_{\sigma(i)} = z_{\sigma(i)}$.

Hence, we always continue to Stage 3 if $f \in C$, and whenever we continue to Stage 3 we can compute all ν_i (for $i \in I$) via self-correction.

2.3.3 Stage 3: Emulating a tester of C'

Recall that when reaching this stage, we may assume that $h = h_I$ is $\epsilon/3$-close to f (w.r.t $\mathcal{D}$). Also recall that $h_I(x)$ depends only on x_{X_I}, where $X_I = \cup_{i\in I} X_i$, and that by Stage 2 we may assume that $\nu_i(z) = z_{\sigma(i)}$ (for every $i \in I$ and almost all z). In light of the forgoing, we define $F : \{0,1\}^n \to \{0,1\}$ such that $F(x) = h(x')$ where $x'_{X_i} = (x_{\sigma(i)}, \ldots, x_{\sigma(i)})$ (i.e.,

[5] By Step 1, we have $h_{S'\cup I}(u) \neq h_{S''\cup I}(u)$, for $S' = [n] \setminus I$ and $S'' = \emptyset$, and in each iteration we cut $S' \setminus S''$ by half while maintaining $h_{S'\cup I}(u) \neq h_{S''\cup I}(u)$.

$x'_j = (x_{X_i})_{\sigma(i)} = x_{\tau(i)}$ if $j \in X_i$)[6] and $x'_j = 0$ otherwise. (Indeed, if $f \in C$, then $F(x) = h(x)$, since $h(y)$ depends only on $(y_{\tau(i)})_{i \in I}$. Using hypothesis that C' (and so C) is closed under zero projection, it follows that $F \in C$.)

We observe that if F is $\epsilon/3$-close (w.r.t $\mathcal{D}$) to both h and C, then f must be ϵ-close to C (since f is $\epsilon/3$-close to h). Hence, we test both these conditions. Specifically, using our ability to sample $\mathcal{D}$, query f, and determine the value of the influential variables in X_I, we proceed as follows:

1. Test whether $F = h$, where testing is w.r.t the distribution $\mathcal{D}$ and proximity parameter $\epsilon/3$.

   This is done by taking $O(1/\epsilon)$ samples of $\mathcal{D}$, and comparing the values of F and h on these sample points. Recall that $h(u) = h_I(u) = f(u_{X_I} \circ 0_{\overline{X_I}})$.

   The value of F on u is determined as follows.

   a. For every $i \in I$, if ν_i is a dictatorship, then set v_i to equal the self-corrected value of $\nu_i(u_{X_i})$, where ν_i is as defined in Stage 2. Otherwise (i.e., when ν_i is an anti-dictatorship), we set v_i to equal the self-corrected value of $\nu_i(u_{X_i}) + 1$.

   b. Return the value $h(u')$, where $u'_j = v_i$ if $j \in X_i$ and $u'_j = 0$ otherwise.

   Indeed, $F = h$ always passes this test, whereas F that is $\epsilon/3$-far from h (w.r.t $\mathcal{D}$) is rejected w.h.p.
2. Test whether F is in C, where testing is w.r.t the distribution $\mathcal{D}$ and proximity parameter $\epsilon/3$. This is done by testing whether F' is in C, where $F'(z) = F(x)$ such that $x_j = z_i$ if j is in the i^{th} set in the collection I, and $x_j = 0$ otherwise. Here we use a distribution-free tester, and analyze it w.r.t the distribution $\mathcal{D}_I$. Toward this end, we need to samples $\mathcal{D}_I$ as well as answer queries to F', where both tasks can be performed as in the prior step. Recall that if $f \in C$, then $F \in C$, and this test will accept (w.h.p.), whereas if F is $\epsilon/3$-far from C the test will reject (w.h.p.).

We conclude that if we reached Stage 3 and $f \in C$ (resp., f is ϵ-far from C), then we accept (resp., reject) w.h.p.

2.4 Digest: Our approach vs the original one [24]

Our new approach differs from the original approach of Diakonikolas et al. [24] in two main aspects:

1. In [24], sets that contain influential variables are identified according to their influence, which is defined with respect to the uniform distribution. This definition seems inadequate when dealing with arbitrary distributions. Instead, we identify such a set by searching for two assignments that differ only on this set and yield different function values. The actual process is iterative and places additional constraints on these assignments (as detailed in Section 2.3.1).
2. In [24], given an assignment to the function, the value of the unique influential variable that resides in a given set S is determined by approximating the influence of two subsets of S (i.e., the subsets of locations assigned the value 0 and 1, respectively). In contrast, we determines this value by defining an auxiliary function, which depends on the unknown influential variable, and evaluating this function (via self-correction w.r.t the uniform distribution (!); see Section 2.3.2).

[6] In general, $\tau(i)$ denotes the location in $[n]$ of the $\sigma(i)^{\text{th}}$ element of X_i.

2.5 The Model

In this subsection, we define the testing and learning models.

In the testing model, we consider the problem of testing a Boolean function class C in the uniform and distribution-free testing models. In the distribution-free testing model (resp. uniform model), the algorithm has access to a Boolean function f via a black-box that returns $f(x)$ when a string x is queried. We call this query *membership query* (MQ_f or just MQ). The algorithm also has access to unknown distribution $\mathcal{D}$ (resp. uniform distribution) via an oracle that returns $x \in \{0,1\}^n$ chosen randomly according to the distribution $\mathcal{D}$ (resp. according to the uniform distribution). We call this query *example query* ($\mathrm{ExQ}_{\mathcal{D}}$ (resp. ExQ or ExQ_U)).

A *distribution-free testing algorithm*, [32], (resp. *testing algorithm*) $\mathcal{A}$ for C is an algorithm that, given as input a distance parameter ϵ and the above two oracles to a Boolean function f,

1. if $f \in C$ then $\mathcal{A}$ outputs "accept" with probability at least $2/3$.
2. if f is ϵ-far from every $g \in C$ with respect to the distribution $\mathcal{D}$ (resp. uniform distribution) then $\mathcal{A}$ outputs "reject" with probability at least $2/3$.

We will also call $\mathcal{A}$ *a tester (or ϵ-tester) for the class C* and an algorithm for *ϵ-testing C*.

We say that $\mathcal{A}$ is *one-sided* if it always accepts when $f \in C$; otherwise, it is called *two-sided* algorithm. The *query complexity of $\mathcal{A}$* is the maximum number of queries $\mathcal{A}$ makes on any Boolean function f.

In the learning models, C is a class of representations of Boolean functions rather than a class of Boolean functions. Therefore, we may have two different representations in C that are logically equivalent. In this paper, we assume that this representation is verifiable; that is, given a representation g, one can decide in polynomial time on the length of this representation if $g \in C$.

A *distribution-free proper learning algorithm* (resp. proper learning algorithm under the uniform distribution) $\mathcal{A}$ for C is an algorithm that, given as input an accuracy parameter ϵ, a confidence parameter δ and an access to both MQ_f for the *target function* $f \in C$ and $\mathrm{ExQ}_{\mathcal{D}}$, with unknown $\mathcal{D}$, (resp. ExQ or ExQ_U), with probability at least $1-\delta$, $\mathcal{A}$ returns $h \in C$ that is ϵ-close to f with respect to $\mathcal{D}$ (resp. with respect to the uniform distribution). This model is also called *proper PAC-learning with membership queries* under any distribution (resp. under the uniform distribution) [3, 48].

3 The Distribution-Free Tester

In this section, we sketch the proof of the tester from Section 2.

For a class C of n-bit long Boolean functions and a set $Y = \{y_1, \ldots, y_q\}$ we define $C^*(Y)$ the class of all q-bit long Boolean functions $f(y_1, \ldots, y_q) = g(y_1, \ldots, y_q, 0, \ldots, 0)$ where $g \in C$. We define $C(Y) \subseteq C^*(Y)$ the class of $f \in C(Y)$ that depends on all the variables in Y. That is, all the variables in Y are influential.

Our main result is

▶ **Theorem 1.** *Let C be a class of n-bit long Boolean functions that is symmetric subclass of k-Junta and is closed under zero projection. Suppose for every $Y = \{y_1, \ldots, y_q\}$ with $q \leq k$, there is a tester T_Y for q-bit Boolean function F such that*

1. *T_Y is a polynomial time two-sided distribution-free (resp. uniform-distribution) adaptive ϵ-tester*
2. *If $F \in C(Y)$ then, with probability at least $1-\delta$, T_Y accepts.*

3. *If F is ϵ-far from every function in $C^*(Y)$ w.r.t $\mathcal{D}$ then, with probability at least $1-\delta$, T_Y rejects.*
4. *T_Y makes $M(\epsilon,\delta)$ MQs and $Q(\epsilon,\delta)$ $ExQ_{\mathcal{D}}$ (resp. ExQ_U).*

Then

1. *There is a polynomial time two-sided distribution-free adaptive algorithm for ϵ-testing C that makes*

$$\tilde{O}\left(M(\epsilon/12,1/24)+kQ(\epsilon/12,1/24)+\frac{k}{\epsilon}\right)$$

   queries.
2. *(resp. There is a polynomial time two-sided uniform-distribution adaptive algorithm for ϵ-testing C that makes*

$$\tilde{O}\left(M(\epsilon/12,1/24)+Q(\epsilon/12,1/24)+\frac{k}{\epsilon}\right)$$

   queries.)

Using Goldreich et. al [32], reduction of testing to proper learning we get

▶ **Theorem 2.** *Let C be a class of n-bit long Boolean functions that is symmetric subclass of k-Junta and is closed under zero projection. Suppose for every $Y=\{y_1,\ldots,y_q\}$ with $q\le k$, there is a polynomial time proper learning algorithm that learns $C^*(Y)$ using $M(\epsilon,\delta)$ MQs and $Q(\epsilon,\delta)$ $ExQ_{\mathcal{D}}$s (resp. ExQ_Us). Then*

1. *There is a polynomial time two-sided distribution-free adaptive algorithm for ϵ-testing C that makes*

$$\tilde{O}\left(M(\epsilon/12,1/24)+kQ(\epsilon/12,1/24)+\frac{k}{\epsilon}\right)$$

   queries.
2. *(resp. There is a polynomial time two-sided uniform-distribution adaptive algorithm for ϵ-testing C that makes*

$$\tilde{O}\left(M(\epsilon/12,1/24)+Q(\epsilon/12,1/24)+\frac{k}{\epsilon}\right)$$

   queries.)

Before we give the proof sketch, we give some applications.

3.1 Some Applications

***k*-Junta:** For the class $C=k$-Junta, $C^*(Y)=q$-Junta and since f is q-bit Boolean function $f\in C^*(Y)$ and the tester T_Y can just returns accept. Then $M=Q=0$ and we get a distribution-free tester for k-Junta that makes $\tilde{O}(k/\epsilon)$.

***k*-Linear:** For the class $C=k$-Linear, the sum (over F_2) of at most k variables, we have $C(Y)=\{y_1+y_2+\cdots+y_q\}$. If f is ϵ-far from $C^*(Y)$ then it is ϵ-far from $y_1+y_2+\cdots,y_q$. We can distinguish between $g=y_1+y_2+\cdots+y_q$ and a function that is ϵ-far from g with $O(1/\epsilon)$ $\mathrm{ExQ}_{\mathcal{D}}$. Then $M=0$ and $Q=O(1/\epsilon)$. So we get a distribution-free tester for k-Linear that makes $O(k/\epsilon)$ queries.

***k*-Term:** For the class $C = k$-Term, the conjunction of at most k literals, we have $C(Y) = \{z_1 \wedge z_2 \wedge \cdots \wedge z_q \mid z_i \in \{y_i, \overline{y_i}\}\}$. The tester T_Y asks $O(1/\epsilon)$ $\mathrm{ExQ}_{\mathcal{D}}$. If for all the strings f is zero then the tester accept. Otherwise, there is a string a such that $f(a) = 1$. Then $z_i = y_i$ if $a_i = 1$ and $z_i = \overline{y_i}$ if $a_i = 0$. Then as above, it tests if f is ϵ-far from $z_1 \wedge z_2 \wedge \cdots \wedge z_q$ and we get a distribution-free tester for k-Term that makes $O(k/\epsilon)$ queries.

***s*-term monotone *r*-DNF:** For the class s-term monotone r-DNF (a DNF that contains at most s r-Terms), in the full paper, we give an algorithm that properly learns this class in polynomial time and makes $O(s/\epsilon)$ $\mathrm{ExD}_{\mathcal{D}}$ and $O(rs\log(ns))$ MQs. Since the number of influential variables in any s-term monotone r-DNF is at most sr we have $q \leq k = sr$. Therefore the class $C^*(Y)$ $(n = q \leq sr)$ can be properly learned using $O(s/\epsilon)$ $\mathrm{ExD}_{\mathcal{D}}$ and $O(rs\log(rs))$ MQs. By Theorem 2 we get

▶ **Theorem 3.** *For any $\epsilon > 0$, there is a polynomial time two-sided distribution-free adaptive algorithm for ϵ-testing s-Term Monotone r-DNF that makes $\tilde{O}(rs^2/\epsilon)$ queries.*

In the full paper, we also show that

▶ **Theorem 4.** *For any $\epsilon > 0$, there is a polynomial time two-sided distribution-free adaptive algorithm for ϵ-testing s-Term Unate[7] r-DNF that makes $\tilde{O}(rs^2/\epsilon)$ queries.*

3.2 Proof Sketch

Algorithm 1 A distribution-free tester for subclasses C of k-Junta.

Tester$C(f, \mathcal{D}, \epsilon)$
Input: Oracle that accesses a Boolean function f and $\mathcal{D}$.
Output: If any one of the procedures rejects
then "reject", otherwise, "accept"

1. $(X, V, I) \leftarrow$**ApproxTarget**$(f, \mathcal{D}, \epsilon, 1/3)$.
2. Define $h(x) = f(x_X \circ 0_{\overline{X}})$.
3. **TestSets**(h, X, V, I).
4. Define F
5. **Close**$fF(f, \mathcal{D}, \epsilon, 1/15)$
6. Run T_Y on F
7. Accept

We now sketch the proof of Theorem 1. See the full proof in [13]. The tester is **Tester**C in Algorithm 1. First, **Tester**C calls the procedure **ApproxTarget**, in Algorithm 2. This procedure executes the first stage of the tester. See Subsection 2.3.1. The reason that here the procedure is more complex is because, for learning classes of unate functions the procedure in Algorithm 1 returns witnesses for $h(x) = f(x_{X_I} \circ 0_{\overline{X_I}})$ and not for f (as in Subsection 2.3.1). So those witnesses also give us the unateness of each variable in the function. This is, for example, how we get the result in Theorem 4.

Throughout the paper we denote $X = X_I$ and $h(x) = f(x_X \circ 0_{\overline{X}})$. In [13] we prove the following. The proof sketch is in Subsection 2.3.1.

[7] The function f is unate if there is a such that $f(x + a)$ is monotone.

Algorithm 2 A procedure that finds influential sets $\{X_i\}_{i\in I}$ of f and a witness $v^{(i)}$ for each influential set X_i for $h := f(x_{X_I} \circ 0_{\overline{X_I}})$ where $X_I = \cup_{i\in I} X_i$. Also, whp, h is $(\epsilon/3)$-close to the target.

ApproxTarget$(f, \mathcal{D}, \epsilon, c)$
Input: Oracle that accesses a Boolean function f and
an oracle that draws $x \in \{0,1\}^n$ according to the distribution $\mathcal{D}$.
Output: Either "reject" or (X_I, V, I)

Partition $[n]$ **into** r **sets**
1. Set $r = 2k^2$.
2. Choose uniformly at random a partition $X_1, X_2, \ldots, X_r$ of $[n]$

Find a close function and influential sets
3. Set $X_I = \emptyset$; $I = \emptyset$; $V = \emptyset$; $t(X_I) = 0$.
4. Repeat $M = 3k\ln(15k)/\epsilon$ times
5. Choose $u \in \mathcal{D}$.
6. $t(X_I) \leftarrow t(X_I) + 1$
7. If $f(u_{X_I} \circ 0_{\overline{X_I}}) \neq f(u)$ then
8. $W \leftarrow \emptyset$.
9. $(\ell, w^{(\ell)}) \leftarrow$ Binary Search to find a new influential set X_ℓ using u and $u_{X_I} \circ 0_{\overline{X_I}}$ and a string
10. $w^{(\ell)} \in \{0,1\}^n$ such that $f(w^{(\ell)}) \neq f(w^{(\ell)}_{\overline{X_\ell}} \circ 0_{X_\ell})$;
11. $X_I \leftarrow X_I \cup X_\ell$; $I \leftarrow I \cup \{\ell\}$.
12. If $|I| > k$ then Output("reject").
13. $W = W \cup \{w^{(\ell)}\}$.
14. Choose $w^{(r)} \in W$.
15. If $f(w^{(r)}_{X_I} \circ 0_{\overline{X_I}}) \neq f(w^{(r)}_{X_I \setminus X_r} \circ 0_{\overline{X_I} \cup X_r})$ then $W \leftarrow W \setminus \{w^{(r)}\}$; $v^{(r)} \leftarrow w^{(r)}_{X_I} \circ 0_{\overline{X_I}}$; $V \leftarrow V \cup \{v^{(r)}\}$; If $W \neq \emptyset$ then Goto 14
16. Else If $f(w^{(r)}_{X_I} \circ 0_{\overline{X_I}}) \neq f(w^{(r)})$ then $u \leftarrow w^{(r)}$; Goto 9
17. Else $u \leftarrow w^{(r)}_{\overline{X_r}} \circ 0_{X_r}$; Goto 9
18. $t(X_I) = 0$.
19. If $t(X_I) = c\ln(15k)/\epsilon$ then Output(X_I, V, I).

▶ **Lemma 5.** *Consider steps 1-2 in the* ***ApproxTarget****. If f is a k-junta then, with probability at least $2/3$, for each $i \in [r]$, the set $x(X_i) = \{x_j | j \in X_i\}$ contains at most one influential variable of f.*

▶ **Lemma 6.** *If* ***ApproxTarget*** *does not reject then it outputs (X, V, I) that satisfies*
1. $q = |I| \leq k$.
2. *For every $\ell \in I$, $v^{(\ell)}_{\overline{X}} = 0$ and $f(v^{(\ell)}) \neq f(0_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$. That is, $v^{(\ell)} \in V$ is a witness of $h(x) = f(x_X \circ 0_{\overline{X}})$ for X_ℓ .*
3. *Each $x(X_\ell)$, $\ell \in I$, contains at least one influential variable of $h(x) = f(x_X \circ 0_{\overline{X}})$.*
4. *With probability at least $14/15$*

$$\boldsymbol{Pr}_{u\in\mathcal{D}}[h(x) \neq f(x)] \leq \epsilon/3.$$

▶ **Lemma 7.** *If f is k-junta and each $x(X_i)$ contains at most one influential variable of f then*

1. ***ApproxTarget*** *outputs (X, V, I).*
2. *Each $x(X_\ell)$, $\ell \in I$, contains exactly one influential variable in $h(x) = f(x_X \circ 0_{\overline{X}})$.*
3. *For every $\ell \in I$, $h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}}) = f(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is a literal.*

▶ **Lemma 8.** *The procedure **ApproxTarget** makes $O((k \log k)/\epsilon) = \tilde{O}(k/\epsilon)$ queries.*

The tester then defines $h(x) = f(x_X \circ 0_{\overline{X}})$. Now, since $\mathbf{Pr}_{u \in \mathcal{D}}[h(x) \neq f(x)] \leq \epsilon/3$ it is enough to distinguish between h is in C and h that is $(2\epsilon/3)$-far from every function in C with respect to $\mathcal{D}$.

The tester then moves to the second stage. See Subsection 2.3.2. First, it calls the procedure **TestSets**. See Algorithm 3 in Appendix A. The procedure tests if every $h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is close to a literal. In the procedure, $\mathbf{UniformJunta}(g, k, \epsilon, \delta)$ is Blais's uniform-distribution one-sided tester [7] for k-Junta. For $k = 1$ it tests whether g is a literal or constant function or ϵ-far from any literal and constant function with respect to the uniform distribution.

The following is very easy to prove

▶ **Lemma 9.** *We have*

1. *If h is k-junta and each $x(X_i)$ contains at most one influential variable of f then **TestSets** returns "OK".*
2. *If for some $\ell \in I$, $h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is $(1/30)$-far from every literal with respect to the uniform distribution then, with probability at least $1 - (1/15)$, **TestSets** rejects.*
3. *The procedure **TestSets** makes $O(k)$ queries.*

This test does not give $\tau(i)$ (the index of the influential variable in X_i) but the fact that $h(x_{X_i} \circ v^{(i)}_{\overline{X_i}})$ is close to $x_{\tau(i)}$ or $\overline{x_{\tau(i)}}$ can be used to find the value of $u_{\tau(i)}$ in every assignment $u \in \{0,1\}^n$ without knowing $\tau(i)$. The latter is done, whp, by the procedure **RelVarValues** that uses self-correction. See Algorithm 4 in Appendix A. We have

▶ **Lemma 10.** *We have*

1. *If h is k-Junta and each $x(X_i)$ contains at most one influential variable of h then **RelVarValues** outputs z such that $z_\ell = w_{\tau(\ell)}$ where $h(x_{X_\ell} \circ 0_{\overline{X_\ell}}) \in \{x_{\tau(\ell)}, \overline{x_{\tau(\ell)}}\}$.*
2. *If for every $\ell \in I$ the function $h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is $(1/30)$-close to a literal in $\{x_{\tau(\ell)}, \bar{x}_{\tau(\ell)}\}$ with respect to the uniform distribution, where $\tau(\ell) \in X_\ell$, and **RelVarValues** does not reject then, with probability at least $1 - \delta$, we have: For every $\ell \in I$, $z_\ell = w_{\tau(\ell)}$.*
3. *The procedure **RelVarValues** makes $O(k \log(k/\delta))$ queries.*

Proof. Since $\nu_i(x) = h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is $(1/30)$-close to a literal in $\{x_{\tau(\ell)}, \bar{x}_{\tau(\ell)}\}$ we have that for a uniform random string z, with probability at least $1/15$ we have $\nu(x + z) + \nu(z) = x_{\tau(\ell)}$. If we repeat this test $O(\log(k/\delta))$ times for every $\ell \in I$, we get a success probability $1 - \delta$. Since $|I| \leq k$, **RelVarValues** makes $O(k \log(k/\delta))$ queries. ◀

We collect all the above events that happens with high probability in the following

▶ **Assumption 11.** *For the rest of this section we assume*

1. *If $f \in C$ then*
   - *$h(x) = f(x_X \circ 0_{\overline{X}}) \in C$.*
   - *Each $x(X_\ell)$, $\ell \in I$ contains exactly one influential variable.*
   - *For every $\ell \in I$, $f(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is a literal in $\{x_{\tau(\ell)}, \overline{x_{\tau(\ell)}}\}$.*

2. *If f is ϵ-far from every function in C then*
   - *$h(x) = f(x_X \circ 0_{\overline{X}})$ is $(\epsilon/3)$-close to f with respect to $\mathcal{D}$ and therefore $h(x)$ is $(2\epsilon/3)$-far from every function in C with respect to $\mathcal{D}$.*
   - *For every $\ell \in I$, $f(x_{X_\ell} \circ v_{\overline{X_\ell}}^{(\ell)})$ is $(1/30)$-close to a literal in $\{x_{\tau(\ell)}, \bar{x}_{\tau(\ell)}\}$ with respect to the uniform distribution.*
3. *For any u, we can get $(u_{\tau(\ell)})_{\ell \in I}$ using $\tilde{O}(k)$ queries.*

We are getting now to the third stage. See Subsection 2.3.3. For a function $f \in C$ we define $\mathrm{Rel}(f)$, the set of all influential variables of f. Let $q = |I|$ and $\Gamma := \{\tau(\ell_1), \ldots, \tau(\ell_q)\}$. Notice that, with the above assumption, if $h \in C$ then $\mathrm{Rel}(h) = \Gamma$. We define the class $C(\Gamma)$ (resp. $C^*(\Gamma)$), the set of functions in C with $\mathrm{Rel}(f) = \{x_\gamma \mid \gamma \in \Gamma\}$ (resp. $\mathrm{Rel}(f) \subseteq \{x_\gamma \mid \gamma \in \Gamma\}$). Since C is symmetric and closed under zero projection $C(\Gamma)$ (resp. $C^*(\Gamma)$) is the set of all functions $f(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)}, 0, 0, \ldots, 0)$ where $f \in C$ and $\mathrm{Rel}(f) = \{x_1, \ldots, x_q\}$ (resp $\mathrm{Rel}(f) \subseteq \{x_1, \ldots, x_q\}$). We recall that, for $Y = \{y_1, \ldots, y_q\}$, $C(Y)$ (resp. $C^*(Y)$) are the set of all functions $f(y_1, \ldots, y_q, 0, 0, \ldots, 0)$ where $f \in C$ and $\mathrm{Rel}(f) = \{x_1, \ldots, x_q\}$ (resp. $\mathrm{Rel}(f) \subseteq \{x_1, \ldots, x_q\}$).

Let $F(y_1, \ldots, y_q) = h(z)(= f(z))$ where $z = (y_1)_{X_{\ell_1}} \circ \cdots \circ (y_q)_{X_{\ell_q}} \circ 0_{\overline{X}}$. That is, for every $\ell_i \in I$, $j \in X_{\ell_i}$ we have $z_j = y_i$ and for every $j \in \overline{X}$ we have $z_j = 0$. Then, by Assumption 11, it is easy to prove that (see Subsection 2.3.3)

▶ **Lemma 12.** *We have*

1. *If $h \in C$ then $F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)}) = h$.*
2. *If $h \in C$ then $F(y_1, \ldots, y_q) \in C(Y)$.*
3. *If h is $2\epsilon/3$-far from every function in C with respect to D then either*
   a. *h is $\epsilon/3$-far from $F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)})$ with respect to $\mathcal{D}$,*
   or,
   b. *$F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)})$ is $\epsilon/3$-far from every function in C with respect to $\mathcal{D}$.*

Therefore, we need to do two tests. The first is to distinguish between $h = F$ and h that is $\epsilon/3$-far from F w.r.t $\mathcal{D}$. The second is to distinguish between $F \in C$ and F that is $\epsilon/3$-far from every function in C w.r.t $\mathcal{D}$.

The following result shows that we can query $F(y_1, \ldots, y_q)$ and for every $x \in \{0,1\}^n$ we can extract x_Γ. So, in particular, we can get a sample according to $\mathcal{D}_\Gamma$ and query $F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)})$. The proof is immediate (See Appendix A)

▶ **Lemma 13.** *For the function F we have*

1. *Given $(y_1, \ldots, y_q)$, computing $F(y_1, \ldots, y_q)$ can be done with one query to f.*
2. *Given $x \in \{0,1\}^n$ and δ, there is an algorithm that makes $O(k \log(k/\delta))$ queries and, with probability at least $1 - \delta$, either discovers that $f \notin C$ and then reject or computes $x_\Gamma = (x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)})$ and $F(x_\Gamma)$.*
3. *There is a polynomial time algorithm that makes $O(k \log(k/\delta))$ queries and with probability at least $1 - \delta$ returns a string u in $\{0,1\}^q$ according to the distribution $\mathcal{D}_\Gamma$ and $F(u)$.*
4. *There is a polynomial time algorithm that makes one query and returns a string u in $\{0,1\}^q$ according to the uniform distribution and $F(u)$.*

We now give the procedure **Close**fF that tests whether $h(x) = f(x_X \circ 0_{\overline{X}})$ is $(\epsilon/3)$-far from F with respect to $\mathcal{D}$. See Algorithm 5 and the proof in Appendix A.

▶ **Lemma 14.** *For any ϵ, a constant δ, and (X, V, I) that satisfies Assumption 11, procedure **Close**fF makes $O((k/\epsilon)\log(k/\epsilon)) = \tilde{O}(k/\epsilon)$ queries and*

1. *If $f \in C$ then **Close**fF returns OK.*
2. *If $h(x)$ is $(\epsilon/3)$-far from F with respect to $\mathcal{D}$ then, with probability at least $1-\delta$, **Close**fF rejects.*

The second test is to distinguish between $F \in C$ and F that is $\epsilon/3$-far from C w.r.t the distribution $\mathcal{D}$.

Consider the tester T_Y in Theorem 1. If $h(x) = F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)}) \in C$ then, by Assumption 11, since each $x(X_\ell)$, $\ell \in I$, contains exactly one influential variable, $F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)}) \in C(\Gamma)$. Therefore $F = F(y_1, \ldots, y_q) \in C(Y)$. If $F(x_{\tau(\ell_1)}, \ldots, x_{\tau(\ell_q)})$ is $\epsilon/3$-far from every function in C w.r.t $\mathcal{D}$, then it is $\epsilon/3$-far from every function in $C^*(\Gamma)$ w.r.t $\mathcal{D}_\Gamma$. Therefore, $F = F(y_1, \ldots, y_q)$ is $\epsilon/3$-far from every function in $C^*(Y)$ w.r.t $\mathcal{D}_\Gamma$. Therefore, T_Y can be used for the second test of $F(x_\Gamma) \in C$ vs. $\epsilon/3$ far from every function in C.

Now by, Lemma 13, every MQ to F can be simulated with one MQ to f and every $\mathrm{ExQ}_{\mathcal{D}_\Gamma}$ can be simulated with $O(k \log(k/\delta'))$ $\mathrm{ExQ}_{\mathcal{D}}$ queries and have success probability $1-\delta'$. Since T_Y asks $Q(\epsilon, \delta)$ queries, we need $O(k \log(kQ(\epsilon, \delta)/\delta))$ $\mathrm{ExQ}_{\mathcal{D}}$ to have success $1-\delta$ for all the queries. When $\mathcal{D}$ is uniform then, by Lemma 13, every ExQ_{U_Γ} can be simulated with one query to MQ.

Notice that the success probability in all the procedures above is $1-\delta$ for any constant δ. By choosing $\delta = 1/20$ for each procedure, we get a tester with confidence of at least $2/3$. By Lemmas 8, 9 and the above analysis the query complexity of the tester is as stated in Theorem 1.

4 The Second Tester: Uniform-Distribution Tester for Classes that are Close to k-Junta w.r.t the Uniform Distribution

The second tester in this paper tests classes that are close to k-Junta w.r.t the uniform distribution, that is, for every $f \in C$ and every ϵ, there is k such that every function in C is ϵ-close to some function in k-Junta.

To understand the intuition behind the second tester, we demonstrate it for testing s-term DNF, the class of DNF with at most s terms. This class is close to $(s \log(s/\epsilon))$-Junta. This is because, for every s-term DNF and every ϵ, the function g that results by removing the terms of size greater than or equal to $\log(s/\epsilon)$ in f is ϵ-close to f and $g \in (s \log(s/\epsilon))$-Junta.

Let f be an s-term DNF. Since, for ϵ-testing, the variables that are influential in f are variables in terms of size $d = O(\log(s/\epsilon))$, there are at most $k = sd = O(s \log(s/\epsilon))$ influential variables in f. Suppose we uniformly at random distribute the variables of f into $r = 10k$ bins $X_1, \ldots, X_r$. The influential variables falls into at most k bins. We call those bins influential bins. Terms in f of size greater than d, with high probability, $d/2$ of their variables falls into the uninfluential bins.

We try as before to find the influential bins, but this time, we use uniform random strings in our search. This is because, when we use uniform random strings, with high probability, all the large terms in f are equal zero for those strings, and therefore, no uninfluential bin is found by the search procedure.

We find enough influential bins such that if we substitute a random assignment in the variables of the uninfluential bins, w.h.p., we get a function H that is $\epsilon/4$-close to f. The next key idea is: as we said before, if we have a large term in f, then with high probability, many of its variables fall into the uninfluential bins. So when we substitute a random assignment for the variables in the uninfluential bins, with high probability, the large terms in f vanish in H. Therefore, with high probability, H is $\epsilon/4$ close to f and contains small terms (terms of size at most $d = O(\log(s/\epsilon))$. Since H is s-term d-DNF, it is a function in sd-Junta, and we can use the first tester to test H.

See more details in Appendix B.

References

1. Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing low-degree polynomials over GF(2). In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques, 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003, Proceedings*, pages 188–199, 2003. doi:10.1007/978-3-540-45198-3_17.
2. Noga Alon, Tali Kaufman, Michael Krivelevich, Simon Litsyn, and Dana Ron. Testing reed-muller codes. *IEEE Trans. Information Theory*, 51(11):4032–4039, 2005. doi:10.1109/TIT.2005.856958.
3. Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
4. Roksana Baleshzar, Meiram Murzabulatov, Ramesh Krishnan S. Pallavoor, and Sofya Raskhodnikova. Testing unateness of real-valued functions. *CoRR*, abs/1608.07652, 2016. arXiv:1608.07652.
5. Aleksandrs Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1021–1032, 2016. doi:10.1145/2897518.2897567.
6. Arnab Bhattacharyya, Swastik Kopparty, Grant Schoenebeck, Madhu Sudan, and David Zuckerman. Optimal testing of reed-muller codes. In *Property Testing - Current Research and Surveys*, pages 269–275. Springer, 2010. doi:10.1007/978-3-642-16367-8_19.
7. Eric Blais. Testing juntas nearly optimally. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 151–158, 2009. doi:10.1145/1536414.1536437.
8. Eric Blais, Joshua Brody, and Kevin Matulef. Property testing lower bounds via communication complexity. In *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*, pages 210–220, 2011. doi:10.1109/CCC.2011.31.
9. Eric Blais and Daniel M. Kane. Tight bounds for testing k-linearity. 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*, pages 435–446, 2012. doi:10.1007/978-3-642-32512-0_37.
10. Avrim Blum. Learning a function of r relevant variables. In *Computational Learning Theory and Kernel Machines, 16th Annual Conference on Computational Learning Theory and 7th Kernel Workshop, COLT/Kernel 2003, Washington, DC, USA, August 24-27, 2003, Proceedings*, pages 731–733, 2003. doi:10.1007/978-3-540-45167-9_54.
11. Avrim Blum and Pat Langley. Selection of relevant features and examples in machine learning. *Artif. Intell.*, 97(1-2):245–271, 1997. doi:10.1016/S0004-3702(97)00063-5.
12. Manuel Blum, Michael Luby, and Ronitt Rubinfeld. Self-testing/correcting with applications to numerical problems. *J. Comput. Syst. Sci.*, 47(3):549–595, 1993. doi:10.1016/0022-0000(93)90044-W.
13. Nader H. Bshouty. Almost optimal distribution-free junta testing. *CoRR*, abs/1901.00717, 2018.
14. Nader H. Bshouty and Areej Costa. Exact learning of juntas from membership queries. *Theor. Comput. Sci.*, 742:82–97, 2018. doi:10.1016/j.tcs.2017.12.032.
15. Deeparnab Chakrabarty and C. Seshadhri. A $o(n)$ monotonicity tester for boolean functions over the hypercube. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 411–418, 2013. doi:10.1145/2488608.2488660.
16. Deeparnab Chakrabarty and C. Seshadhri. A $\tilde{O}(n)$ non-adaptive tester for unateness. *CoRR*, abs/1608.06980, 2016. arXiv:1608.06980.

17 Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Efficient sample extractors for juntas with applications. In *Automata, Languages and Programming - 38th International Colloquium, ICALP 2011, Zurich, Switzerland, July 4-8, 2011, Proceedings, Part I*, pages 545–556, 2011. doi:10.1007/978-3-642-22006-7_46.

18 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 519–528, 2015. doi:10.1145/2746539.2746570.

19 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. *CoRR*, abs/1412.5655, 2014. arXiv:1412.5655.

20 Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond talagrand functions: new lower bounds for testing monotonicity and unateness. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 523–536, 2017. doi:10.1145/3055399.3055461.

21 Xi Chen, Erik Waingarten, and Jinyu Xie. Boolean unateness testing with $\widetilde{O}(n^{3/4})$ adaptive queries. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 868–879, 2017. doi:10.1109/FOCS.2017.85.

22 Xi Chen and Jinyu Xie. Tight bounds for the distribution-free testing of monotone conjunctions. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 54–71, 2016. doi:10.1137/1.9781611974331.ch5.

23 Peter Damaschke. Adaptive versus nonadaptive attribute-efficient learning. *Machine Learning*, 41(2):197–215, 2000. doi:10.1023/A:1007616604496.

24 Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Krzysztof Onak, Ronitt Rubinfeld, Rocco A. Servedio, and Andrew Wan. Testing for concise representations. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2007), October 20-23, 2007, Providence, RI, USA, Proceedings*, pages 549–558, 2007. doi:10.1109/FOCS.2007.32.

25 Ilias Diakonikolas, Homin K. Lee, Kevin Matulef, Rocco A. Servedio, and Andrew Wan. Efficiently testing sparse $GF(2)$ polynomials. *Algorithmica*, 61(3):580–605, 2011. doi:10.1007/s00453-010-9426-9.

26 Elya Dolev and Dana Ron. Distribution-free testing for monomials with a sublinear number of queries. *Theory of Computing*, 7(1):155–176, 2011. doi:10.4086/toc.2011.v007a011.

27 Eldar Fischer, Guy Kindler, Dana Ron, Shmuel Safra, and Alex Samorodnitsky. Testing juntas. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Vancouver, BC, Canada, Proceedings*, pages 103–112, 2002. doi:10.1109/SFCS.2002.1181887.

28 Dana Glasner and Rocco A. Servedio. Distribution-free testing lower bound for basic boolean functions. *Theory of Computing*, 5(1):191–216, 2009. doi:10.4086/toc.2009.v005a010.

29 Oded Goldreich, editor. *Property Testing - Current Research and Surveys*, volume 6390 of *Lecture Notes in Computer Science*. Springer, 2010. doi:10.1007/978-3-642-16367-8.

30 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017. doi:10.1017/9781108135252.

31 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000. doi:10.1007/s004930070011.

32 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

33 Parikshit Gopalan, Ryan O'Donnell, Rocco A. Servedio, Amir Shpilka, and Karl Wimmer. Testing fourier dimensionality and sparsity. *SIAM J. Comput.*, 40(4):1075–1100, 2011. doi:10.1137/100785429.

34 David Guijarro, Jun Tarui, and Tatsuie Tsukiji. Finding relevant variables in PAC model with membership queries. In *Algorithmic Learning Theory, 10th International Conference, ALT '99, Tokyo, Japan, December 6-8, 1999, Proceedings*, page 313, 1999. doi:10.1007/3-540-46769-6_26.

35 Shirley Halevy and Eyal Kushilevitz. Distribution-free property-testing. *SIAM J. Comput.*, 37(4):1107–1138, 2007. doi:10.1137/050645804.

36 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and boolean isoperimetric type theorems. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 52–58, 2015. doi:10.1109/FOCS.2015.13.

37 Subhash Khot and Igor Shinkar. An $\widetilde{O}(n)$ queries adaptive tester for unateness. *CoRR*, abs/1608.02451, 2016. arXiv:1608.02451.

38 Richard J. Lipton, Evangelos Markakis, Aranyak Mehta, and Nisheeth K. Vishnoi. On the fourier spectrum of symmetric boolean functions with applications to learning symmetric juntas. In *20th Annual IEEE Conference on Computational Complexity (CCC 2005), 11-15 June 2005, San Jose, CA, USA*, pages 112–119, 2005. doi:10.1109/CCC.2005.19.

39 Zhengyang Liu, Xi Chen, Rocco A. Servedio, Ying Sheng, and Jinyu Xie. Distribution-free junta testing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 749–759, 2018. doi:10.1145/3188745.3188842.

40 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing ± 1-weight halfspace. In *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*, pages 646–657, 2009. doi:10.1007/978-3-642-03685-9_48.

41 Kevin Matulef, Ryan O'Donnell, Ronitt Rubinfeld, and Rocco A. Servedio. Testing halfspaces. *SIAM J. Comput.*, 39(5):2004–2047, 2010. doi:10.1137/070707890.

42 Elchanan Mossel, Ryan O'Donnell, and Rocco A. Servedio. Learning functions of k relevant variables. *J. Comput. Syst. Sci.*, 69(3):421–434, 2004. doi:10.1016/j.jcss.2004.04.002.

43 Michal Parnas, Dana Ron, and Alex Samorodnitsky. Testing basic boolean formulae. *SIAM J. Discrete Math.*, 16(1):20–46, 2002. URL: http://epubs.siam.org/sam-bin/dbq/article/40744.

44 Dana Ron. Property testing: A learning theory perspective. *Foundations and Trends in Machine Learning*, 1(3):307–402, 2008. doi:10.1561/2200000004.

45 Dana Ron. Algorithmic and analysis techniques in property testing. *Foundations and Trends in Theoretical Computer Science*, 5(2):73–205, 2009. doi:10.1561/0400000029.

46 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

47 Mert Saglam. Near log-convexity of measured heat in (discrete) time and consequences. In *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 967–978, 2018. doi:10.1109/FOCS.2018.00095.

48 Leslie G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, 1984. doi:10.1145/1968.1972.

A The Procedures of the First Tester

Proof of Lemma 13.

Proof. *1* is immediate since $F(y_1,\ldots,y_q) = f(z)$ where $z = (y_1)_{X_{\ell_1}} \circ \cdots \circ (y_q)_{X_{\ell_q}} \circ 0_{\overline{X}}$. To prove *2*. We run **RelVarValues**(x, X, V, I, δ). If it rejects then, by Lemma 10, $f \notin C$. Otherwise, by Lemma 10, with probability at least $1-\delta$, the procedure outputs z where for every ℓ, $z_\ell = x_{\tau(\ell)}$. Then using *1* we compute $F(z)$. Since by Lemma 10, **RelVarValue** makes $O(k\log(k/\delta))$ queries, the result follows. Now 3 and 4 follows immediately from 1 and 2. ◀

Algorithm 3 A procedure that tests if for all $\ell \in I$, $h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}})$ is (1/30)-close to some literal with respect to the uniform distribution.

TestSets(h, X, V, I)
Input: Oracle that accesses a Boolean function f and (X, V, I).
Output: Either "reject" or "OK"

1. For every $\ell \in I$ do
2. If **UniformJunta**$(h(x_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}}), 1, 1/30, 1/15)$="reject"
3. then Output("reject")
4. Choose $b \in \{0,1\}^n$ uniformly at random
5. If $h(b_{X_\ell} \circ v^{(\ell)}_{\overline{X_\ell}}) = h(\overline{b_{X_\ell}} \circ v^{(\ell)}_{\overline{X_\ell}})$ then Output("reject")
6. Return "OK"

Algorithm 4 A procedure that takes as input (X, V, I) and a string $w \in \{0,1\}^n$ and, with probability at least $1 - \delta$, returns the values of $w_{\tau(i)}$, $i \in I$, where $h(x_{X_i} \circ v^{(i)}_{\overline{X_i}})$ is (1/30)-close to one of the literals in $\{x_{\tau(i)}, \overline{x_{\tau(i)}}\}$ with respect to the uniform distribution.

RelVarValues(w, X, V, I, δ)
Input: Oracle that accesses a Boolean function h, (X, V, I) and $w \in \{0,1\}^n$.
Output: Either "reject" or returns for every $\ell \in I$, the value, $z_\ell = w_{\tau(\ell)}$ where $x_{\tau(\ell)}$ is one of the influential variables of $h(x_X \circ 0_{\overline{X}})$ in $x(X_\ell)$

1. For every $\ell \in I$ do
2. For $\xi \in \{0,1\}$ set $Y_{\ell,\xi} = \{j \in X_\ell | w_j = \xi\}$.
3. Set $G_{\ell,0} = G_{\ell,1} = 0$;
4. Repeat $t = \ln(k/\delta)/\ln(4/3)$ times
5. Choose $b \in \{0,1\}^n$ uniformly at random;
6. If $h(b_{Y_{\ell,0}} \circ b_{Y_{\ell,1}} \circ v^{(\ell)}_{\overline{X_\ell}}) \neq h(\overline{b_{Y_{\ell,0}}} \circ b_{Y_{\ell,1}} \circ v^{(\ell)}_{\overline{X_\ell}})$ then $G_{\ell,0} \leftarrow G_{\ell,0} + 1$
7. If $h(b_{Y_{\ell,1}} \circ b_{Y_{\ell,0}} \circ v^{(\ell)}_{\overline{X_\ell}}) \neq h(\overline{b_{Y_{\ell,1}}} \circ b_{Y_{\ell,0}} \circ v^{(\ell)}_{\overline{X_\ell}})$ then $G_{\ell,1} \leftarrow G_{\ell,1} + 1$
8. If $(\{G_{\ell,0}, G_{\ell,1}\} \neq \{0, h\})$ then Output("reject")
9. If $G_{\ell,0} = t$ then $z_\ell \leftarrow 0$ else $z_\ell \leftarrow 1$
10. Output("$\{z_\ell\}_{\ell \in I}$")

Algorithm 5 A procedure that tests whether $h(x)$ is $(\epsilon/3)$-far from F with respect to $\mathcal{D}$.

Close$fF(f, \mathcal{D}, \epsilon, \delta)$
Input: Oracle that accesses a Boolean function f and $\mathcal{D}$.
Output: Either "reject" or "OK"

1. Repeat $t = (3/\epsilon)\ln(2/\delta)$ times
2. Choose $u \in \mathcal{D}$.
3. $z \leftarrow$**RelVarValue**$(u, X, V, I, \delta/(2t))$.
4. If $h(u) \neq F(z)$ then Output("reject")
5. Return "OK".

Proof of Lemma 14.

Proof. If $f \in C$ then, by Lemma 12, $F(u_\Gamma^{(i)}) = h(u)$ for every i. By Lemma 10 and Assumption 11, $z^{(i)} = u_\Gamma^{(i)}$ for all i, and therefore **Close**fF returns OK.

Suppose now $h(x)$ is $(\epsilon/3)$-far from F with respect to $\mathcal{D}$. By *2* in Lemma 13, **RelVarValue** makes $O(kt\log((kt)/\delta))$ queries and computes $F(u_\Gamma^{(i)})$, $i = 1, \ldots, t$, with failure probability at most $\delta/2$. Then the probability that it fails to reject is at most $(1-\epsilon/3)^t \le \delta/2$. This gives the result.

Therefore, **Close**fF makes $O((k/\epsilon)\log(k/\epsilon))$ queries and satisfies *1* and *2*. ◀

B Classes that are Close to k-Junta

We now give more details. The tester first runs the procedure **Approx**C in Algorithm 6. This procedure is similar to the procedure **ApproxTarget**. It randomly uniformly partitions the variables to $r = 4c^2(c+1)s\log(s/\epsilon)$ disjoint sets $X_1, \ldots, X_r$ and finds influential sets $\{X_i\}_{i\in I}$. Here c is a large constant. To find a new influential set, it chooses two random uniform strings $u, v \in \{0,1\}^n$ and verifies if $f(u_X \circ v_{\overline{X}}) \neq f(u)$ where X is the union of the influential sets that it has found thus far. If $f(u_X \circ v_{\overline{X}}) \neq f(u)$ then the binary search finds a new influential set.

In the binary search for a new influential set, the procedure defines a set X' that is equal to the union of half of the sets in $\{X_i\}_{i\notin I}$. Then either $f(u_{X\cup X'} \circ v_{\overline{X'}}) \neq f(u)$ or $f(u_{X\cup X'} \circ v_{\overline{X'}}) \neq f(u_X \circ v_{\overline{X}})$. Then it recursively does the above until it finds a new influential set X_ℓ.

It is easy to see that if f is s-term DNF then, whp, for all the terms T in f of size at least $c^2\log(s/\epsilon)$, for all the random uniform strings u, v chosen in the algorithm and for all the strings z generated in the binary search, $T(u_X \circ v_{\overline{X}}) = T(u) = T(z) = 0$. Therefore, when f is s-term DNF, the procedure, whp, runs as if there are no terms of size greater than $c^2\log(s/\epsilon)$ in f. This shows that, whp, each influential set that the procedure finds contains at least one variable that belongs to a term of size at most $c^2\log(s/\epsilon)$ in f. Therefore, if f is s-term DNF, the procedure, whp, does not generate more than $c^2 s\log(s/\epsilon)$ influential sets. If the procedure finds more than $c^2 s\log(s/\epsilon)$ influential sets then, whp, f is not s-term DNF and therefore it rejects.

Let R be the set of all the variables that belong to the terms in f of size at most $c^2\log(s/\epsilon)$. The procedure returns $h(x) = f(x_X \circ w_{\overline{X}})$ for random uniform w where X is the union of the influential sets $X = \cup_{i\in I} X_i$ that is found by the procedure. If f is s-term DNF then since $r = 4c^2(c+1)s\log(s/\epsilon)$ and the number of influential sets is at most $c^2 s\log(s/\epsilon)$, whp, at least $(1/2)c\log(s/\epsilon)$ variables in each term of f that contains at least $c\log(s/\epsilon)$ variables not in R falls outside X in the partition of $[n]$. Therefore, for random uniform w, whp, terms T in f that contains at least $c\log(s/\epsilon)$ variables not in R satisfies $T(x_X \circ w_{\overline{X}}) = 0$ and therefore, whp, are vanished in $H = f(x_X \circ w_{\overline{X}})$. Thus, whp, H contains all the terms that contains variables in R and at most $cs\log(s/\epsilon)$ variables not in R. Therefore, whp, H contains at most $c(c+1)s\log(s/\epsilon)$ influential variables. From this, and using similar arguments as for the procedure **ApproxTarget** in the previous subsection, we prove that, **Approx**C makes at most $\tilde{O}(s/\epsilon)$ queries and

1. If f is s-term DNF then, whp, the procedure outputs X and w such that
   - $H = f(x_X \circ w_{\overline{X}})$ is s-term DNF[8].
   - The number of influential variables in $H = f(x_X \circ w_{\overline{X}})$ is at most $O(s\log(s/\epsilon))$.

[8] So in this case we need the class to be closed under zero-one projection.

2. If f is ϵ-far from every s-term DNF then the procedure either rejects or outputs X and w such that, whp, $H = f(x_X \circ w_{\overline{X}})$ is $(3\epsilon/4)$-far from every s-term DNF.

We can now run **Tester**C (with $3\epsilon/4$) on H from the previous subsection for testing C^* where C^* is the set of s-term DNF with $k = O(s \log(s/\epsilon))$ influential variables.

Algorithm 6 A procedure that removes variables from f that only appear in large size terms.

Algorithm Approx$C(f, \epsilon)$
Input: Oracle that accesses a Boolean function f and ϵ
Output: Either "$X \subseteq [n], w \in \{0,1\}^n$" or "reject"

Partition $[n]$ **into** r **sets**
1. Set $r = 8sc\log(s/\epsilon)$.
2. Choose uniformly at random a partition $X_1, X_2, \ldots, X_r$ of $[n]$

Find a close function and influential sets
3. Set $X = \emptyset$; $I = \emptyset$; $t(X) = 0$; $k = 3ms$
4. Repeat $M = 400k\ln(100k)/\epsilon$ times
5. Choose w and u uniformly at random from $\{0,1\}^n$;.
6. $t(X) \leftarrow t(X) + 1$
7. If $f(u_X \circ v_{\overline{X}}) \neq f(u)$ then
8. Binary Search to find a new influential set X_ℓ; $X \leftarrow X \cup X_\ell$; $I \leftarrow I \cup \{\ell\}$.
9. If $|I| > k$ then output "reject" and halt.
10. $t(X) = 0$.
11. If $t(X) = 400\ln(100k)/\epsilon$ then
12. Sample w uniformly at random from $\{0,1\}^n$;
13. Output$(X, w, H = f(x_X \circ w_{\overline{X}}))$.

Palette Sparsification Beyond $(\Delta + 1)$ Vertex Coloring

Noga Alon
Department of Mathematics, Princeton University, NJ, USA
Schools of Mathematics and Computer Science, Tel Aviv University, Israel

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

Abstract

A recent *palette sparsification theorem* of Assadi, Chen, and Khanna [SODA'19] states that in every n-vertex graph G with maximum degree Δ, sampling $O(\log n)$ colors per each vertex independently from $\Delta + 1$ colors almost certainly allows for proper coloring of G from the sampled colors. Besides being a combinatorial statement of its own independent interest, this theorem was shown to have various applications to design of algorithms for $(\Delta + 1)$ coloring in different models of computation on massive graphs such as streaming or sublinear-time algorithms.

In this paper, we focus on palette sparsification beyond $(\Delta + 1)$ coloring, in both regimes when the number of available colors is much larger than $(\Delta+1)$, and when it is much smaller. In particular,

- We prove that for $(1 + \varepsilon)\Delta$ coloring, sampling only $O_\varepsilon(\sqrt{\log n})$ colors per vertex is sufficient and necessary to obtain a proper coloring from the sampled colors – this shows a separation between $(1 + \varepsilon)\Delta$ and $(\Delta + 1)$ coloring in the context of palette sparsification.
- A natural family of graphs with chromatic number much smaller than $(\Delta + 1)$ are triangle-free graphs which are $O(\frac{\Delta}{\ln \Delta})$ colorable. We prove a palette sparsification theorem tailored to these graphs: Sampling $O(\Delta^\gamma + \sqrt{\log n})$ colors per vertex is sufficient and necessary to obtain a proper $O_\gamma(\frac{\Delta}{\ln \Delta})$ coloring of triangle-free graphs.
- We also consider the "local version" of graph coloring where every vertex v can only be colored from a list of colors with size proportional to the degree $\deg(v)$ of v. We show that sampling $O_\varepsilon(\log n)$ colors per vertex is sufficient for proper coloring of any graph with high probability whenever each vertex is sampling from a list of $(1 + \varepsilon) \cdot \deg(v)$ arbitrary colors, or even only $\deg(v) + 1$ colors when the lists are the sets $\{1, \ldots, \deg(v) + 1\}$.

Our new palette sparsification results naturally lead to a host of new and/or improved algorithms for vertex coloring in different models including streaming and sublinear-time algorithms.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Mathematics of computing → Graph coloring; Mathematics of computing → Graph algorithms

Keywords and phrases Graph coloring, palette sparsification, sublinear algorithms, list-coloring

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.6

Category RANDOM

Related Version Full version hosted on arXiv [1]: `https://arxiv.org/abs/2006.10456`.

Funding *Noga Alon*: Supported in part by NSF grant DMS-1855464 and the Simons Foundation. *Sepehr Assadi*: Part of this work was done while the author was a postdoctoral researcher at Princeton University and was supported in part by Simons Collaboration on Algorithms and Geometry.

Acknowledgements Sepehr Assadi would like to thank Suman Bera, Amit Chakrabarti, Prantar Ghosh, Guru Guruganesh, David Harris, Sanjeev Khanna, and Hsin-Hao Su for helpful conversations and Mohsen Ghaffari for communicating the (deg +1) coloring problem and an illuminating discussion that led us to the proof of the palette sparsification theorem for this problem in this paper. We are also thankful to the anonymous reviewers of RANDOM 2020 for helpful suggestions.



© Noga Alon and Sepehr Assadi;
licensed under Creative Commons License CC-BY



1 Introduction

Given a graph $G(V, E)$, let $n := |V|$ be the number of vertices and Δ denote the maximum degree. A proper c-coloring of G is an assignment of colors to vertices from the palette of colors $\{1, \ldots, c\}$ such that adjacent vertices receive distinct colors. The minimum number of colors needed for proper coloring of G is referred to as the chromatic number of G and is denoted by $\chi(G)$. An interesting variant of graph coloring is *list-coloring* whereby every vertex v is given a set $S(v)$ of available colors and the goal is to find a proper coloring of G such that the color of every v belongs to $S(v)$. When this is possible, we say that G is list-colorable from the lists S.

It is well-known that $\chi(G) \leq \Delta + 1$ for every graph G; the algorithmic problem of finding such a coloring – the $(\Delta + 1)$ coloring problem – can also be solved via a text-book greedy algorithm. Very recently, Assadi, Chen, and Khanna [5] proved the following **palette sparsification theorem** for the $(\Delta + 1)$ coloring problem: Suppose for every vertex v of a graph G, we *independently* sample $O(\log n)$ colors $L(v)$ uniformly at random from the palette $\{1, \ldots, \Delta + 1\}$; then G is almost-certainly list-colorable from the sampled lists L.

The palette sparsification theorem of [5], besides being a purely graph-theoretic result of its own independent interest, also had several interesting algorithmic implications for the $(\Delta + 1)$ coloring problem owing to its "sparsification" nature: it is easy to see that by sampling only $O(\log n)$ colors per vertex, the total number of edges that can ever become monochromatic while coloring G from lists L is with high probability only $O(n \cdot \log^2 n)$; at the same time we can safely ignore all other edges of G. This theorem thus reduces the $(\Delta + 1)$ coloring problem, in a *non-adaptive* way, to a list-coloring problem on a graph with (potentially) much smaller number of edges.

The aforementioned aspect of this palette sparsification is particularly appealing for the design of *sublinear algorithms* – these are algorithms which require computational resources that are substantially smaller than the size of their input. Indeed, one of the interesting applications of this theorem, proven (among other things) in [5], is a randomized algorithm for the $(\Delta + 1)$ coloring problem that runs in $\widetilde{O}(n\sqrt{n})$[1] time; for sufficiently dense graphs, this is faster than even reading the entire input once!

Palette sparsification in [5] was tailored to the $(\Delta + 1)$ coloring problem. Motivated by the ubiquity of graph coloring problems on one hand, and the wide range of applications of this palette sparsification result on the other hand, the following question is natural:

What other graph coloring problems admit (similar) palette sparsification theorems?

This is precisely the question we study in this work from both upper and lower bound fronts.

1.1 Our Contributions

We consider palette sparsification beyond $(\Delta + 1)$ coloring: when the number of available colors is much larger than $\Delta + 1$, when it is much smaller, and when the number of available colors for vertices depend on "local" parameters of the graph.

$(1 + \varepsilon)\Delta$ Coloring. The palette sparsification theorem of [5] is shown to be tight in the sense that on some graphs, sampling $o(\log n)$ colors per vertex from $\{1, \ldots, \Delta + 1\}$, results in the sampled list-coloring instance to have no proper coloring with high probability. We prove that in contrast to this, if one allows for a larger number of available colors, then indeed we can obtain a palette sparsification with asymptotically smaller sampled lists.

[1] Here and throughout the paper, we use the notation $\widetilde{O}(f) := O(f \cdot \text{polylog}(f))$ to suppress log-factors.

▶ **Result 1** (Informal – Formalized in Theorem 5). *For any graph $G(V,E)$, sampling $O_\varepsilon(\sqrt{\log n})$ colors per vertex from a set of size $(1+\varepsilon)\Delta$ colors with high probability allows for a proper list-coloring of G from the sampled lists.*

Result 1, combined with the lower bound of [5], provides a separation between $(\Delta+1)$ coloring and $(1+\varepsilon)\Delta$ coloring in the context of palette sparsification. We also prove that the bound of $\Theta(\sqrt{\log n})$ sampled colors is (asymptotically) optimal in Result 1.

To prove Result 1, we unveil a new connection between palette sparsification theorems and some of the classical list-coloring problems in the literature. In particular, several works in the past (see, e.g. [31, 20, 32] and [3, Proposition 5.5.3]) have studied the following question: Suppose in a list-coloring instance on a graph G, we define the c-degree of a vertex-color pair (v,c) as the number of neighbors of v that also contain c in their list; what conditions on maximum c-degrees and minimum list sizes imply that G is list-colorable from such lists?

Palette sparsification theorems turned out to be closely related to these questions as the sampled lists in these results can be viewed through the lens of these list-coloring results. In particular, Reed and Sudakov [32] proved that in the above question if the size of each list is larger than the maximum c-degree by a $(1+o(1))$ factor, then G is always list-colorable. The question here is then whether or not the lists sampled in Result 1 satisfy this condition with high probability. The answer turns out to be *no* as sampling only $O(\sqrt{\log n})$ colors does not provide the proper concentration needed for this guarantee. Despite this, we show that one can still use [32] to prove Result 1 with a more delicate argument by applying [32] to carefully chosen subsets of the sampled lists.

$O(\frac{\Delta}{\ln \Delta})$ Coloring of Triangle-Free Graphs. Even though $\chi(G)$ in general can be $\Delta+1$, many natural families of graphs have chromatic number (much) smaller than $\Delta+1$. One key example is the set of triangle-free graphs which are $O(\frac{\Delta}{\ln \Delta})$ colorable by a celebrated result of Johansson [21] (which was recently simplified and improved to $(1+o(1))\cdot\frac{\Delta}{\ln\Delta}$ by Molloy [24]; see also [29, 8]). We prove a palette sparsification theorem for these graphs.

▶ **Result 2** (Informal – Formalized in Theorem 6). *For any triangle-free graph $G(V,E)$, sampling $O(\Delta^\gamma+\sqrt{\log n})$ colors per vertex from a set of size $O_\gamma(\frac{\Delta}{\ln\Delta})$ colors with high probability allows for a proper list-coloring of G from the sampled lists.*

Unlike Result 1 of our paper and the theorem of [5], in this result we also have a dependence of Δ^γ on the number of sampled colors (where the exponent depends on the number of available colors). We prove that this dependence is also necessary (Proposition 8).

The proof of Result 2 is also based on the aforementioned connection to list-coloring problems based on c-degrees. However, unlike the case for Result 1, here we are not aware of any such list-coloring result that allows us to infer Result 2. As such, a key part of the proof of Result 2 is exactly to establish such a result. Our proof for the corresponding list-coloring problem is by the probabilistic method and in particular a version of the so-called "Rödl Nibble" or the "semi-random method"; see, e.g. [33, 26]. Similar to previous work on coloring triangle-free graphs, the main challenge here is to establish the desired concentration bounds. We do this following the approach of Pettie and Su [29] in their distributed algorithm for coloring triangle-free graphs.

We shall note that our proofs of Results 1 and 2 are almost entirely disjoint from the techniques in [5] and instead build on classical work on list-coloring problems in graph theory.

Coloring with Local Lists Size. Finally, we consider a coloring problem with "local" list sizes where the number of available colors for vertices depends on a local parameter, namely their degree as opposed to a global parameter such as maximum degree.

▶ **Result 3** (Informal – Formalized in Theorem 12). *For any graph $G(V, E)$, sampling $O_\varepsilon(\log n)$ colors for each vertex v with degree $\deg(v)$ from a set $S(v)$ of $(1+\varepsilon)\cdot\deg(v)$ arbitrary colors or only $\deg(v)+1$ colors when the lists are the sets $\{1,\ldots,\deg(v)+1\}$, allows for a proper coloring of G from the sampled colors.*

Coloring problems with local lists size have been studied before in both the graph theory literature, e.g. in [14, 11] for coloring triangle-free graphs (and as pointed out by [14], the general idea goes all the way back to the notion of degree-choosability in one of the original list-coloring papers [16]), and theoretical computer science, e.g. in [13].

To be more precise, the first part of Result 3 refers to the standard $(1+\varepsilon)\deg$ *list*-coloring problem and the second part corresponds to the so-called $(\deg+1)$ coloring problem introduced first (to our knowledge) in the recent work of Chang, Li, and Pettie [13] (see also [4] for an application of this problem). We remark that the $(\deg+1)$ coloring problem is a generalization of the $(\Delta+1)$ coloring problem and hence our Result 3 generalizes that of [5] (although we build on many of the ideas and tools developed in [5] for $\Delta+1$ coloring).

Our proof of Result 3 takes a different route than Results 1 and 2 that were based on list-coloring and instead we follow the approach of [5] for $(\Delta+1)$ coloring. A fundamental challenge here is that the graph decomposition for partitioning vertices into sparse and dense parts that played a key role in [5] is no longer applicable to the $(\deg+1)$ coloring problem. We address this by "relaxing" the requirements of the decomposition and develop a new one that despite being somewhat "weaker" than the ones for $(\Delta+1)$ coloring in [18, 13, 5] (themselves based on [30]), takes into account the disparity between degrees of vertices in the $(\deg+1)$ coloring problem. Similar to [5], we then handle "sparse"[2] and dense vertices of this decomposition separately but unlike [5], here the main part of the argument is to handle these "sparse" vertices and the result for the dense part follows more or less directly from [5].

We conclude this section by noting that our proof for $(1+\varepsilon)\deg$-list coloring problem also immediately gives a palette sparsification result for obtaining a $(1+\varepsilon)\kappa$-list coloring with sampling $O_\varepsilon(\log n)$ colors, where κ is the degeneracy of the graph. This problem was studied very recently in the context of sublinear or "space conscious" algorithms by Bera, Chakrabarti, and Ghosh [7] who also proved, among other interesting results, that $(\kappa+1)$ coloring cannot be achieved via palette sparsification – our result thus complements their lower bound. We postpone the details of this result to the full version of the paper.

1.2 Implication to Sublinear Algorithms for Graph Coloring

As stated earlier, one motivation in studying palette sparsification is in its application to design of sublinear algorithms. As was shown in [5], these theorems imply sublinear algorithms in various models in "almost" a black-box way (see Section 5 for details). For concreteness, in this paper, we stick to their application to the two canonical examples of streaming and sublinear-time algorithms. We only note in passing that exactly as in [5], our results also imply new algorithms in models such as massively parallel computation (MPC) or distributed/linear sketching; see also [12, 7] for more recent results on graph coloring problems in these and related models.

[2] Technically speaking, this decomposition allows for vertices that are neither sparse nor dense and are key to extending the decomposition from $(\Delta+1)$ coloring to $(\deg+1)$ coloring.

Table 1 A sample of our sublinear algorithms as corollaries of Results 1, 2, and 3, and the previous work in [5] and [7] (for brevity, we assume ε, γ are constants). All streaming algorithms here are *single-pass* and all sublinear-time algorithms except for $(1+\varepsilon)\kappa$ coloring are *non-adaptive*.

Problem	Graph Family	Streaming	Sublinear-Time	Source
$(\Delta+1)$ Coloring	General	$O(n \log^2 n)$ space	$\widetilde{O}(n^{3/2})$ time	[5]
$(1+\varepsilon)\kappa$ Coloring	κ-Degenerate	$O(n \log n)$ space	$\widetilde{O}(n^{3/2})$ time	[7]
$O_\gamma(\frac{\Delta}{\ln \Delta})$ Coloring	Triangle-Free	$O(n \cdot \Delta^\gamma)$ space	$O(n^{3/2+\gamma})$ time	our work
$(1+\varepsilon)$ deg List-Coloring	General	$O(n \cdot \log^2 n)$ space	$\widetilde{O}(n^{3/2})$ time	our work
(deg +1) Coloring	General	$O(n \cdot \log^2 n)$ space	$\widetilde{O}(n^{3/2})$ time	our work

Our results in this part appear in Section 5. Table 1 presents a summary of our sublinear algorithms and the directly related previous work (even though our Result 1 implies a separation between $(\Delta+1)$ and $(1+\varepsilon)\Delta$ coloring, the resulting sublinear algorithms from Result 1 are subsumed by the previous work in [7] and hence are omitted from Table 1).

Sublinear Algorithms from Graph Partitioning. Motivated by our results on sublinear algorithms for triangle-free graphs, we also consider sublinear algorithms for coloring other "locally sparse" graphs such as K_r-free graphs, locally r-colorable graphs, and graphs with sparse neighborhood. We give several results for these problems through a general algorithm based on the graph partitioning technique (see, e.g. [12, 27, 28, 7]). Our results in this part are presented in Appendix B.

2 Preliminaries

Notation. For any integer $t \geq 1$, we define $[t] := \{1, \ldots, t\}$. For a graph $G(V, E)$, we use $V(G) := V$ and $E(G) := E$ to denote the vertex-set and edge-set respectively. For a vertex $v \in V$, $N_G(v)$ denotes the neighborhood of v in G and $\deg_G(v) := |N_G(v)|$ denotes the degree of v (when clear from the context, we may drop the subscript G). For a vertex-set $U \subseteq V$, $G[U]$ denotes the induced subgraph of G on U. When there are lists of colors $S(v)$ given to vertices v, we use the term ***c*-degree** of v to mean the number of neighbors u of v of with color c in their list $S(u)$ and denote this by $\deg_S(v, c)$. We use the term "with high probability" (w.h.p.) for an event to mean that the probability of this event happening is at least $1 - 1/n^c$ where c is a sufficiently large constant.

List-Coloring with Constraints on Color-Degrees. We use the following result of Reed and Sudakov [32] on list-coloring of graphs with constraints on c-degrees of vertices.

▶ **Proposition 4** ([32]). *For every $\varepsilon > 0$ there exists a $d_0 := d_0(\varepsilon)$ such that for all $d \geq d_0$ the following is true. Suppose $G(V, E)$ is a graph with lists $S(v)$ for every $v \in V$ such that:*

1. *for every vertex v, $|S(v)| \geq (1+\varepsilon) \cdot d$, and*
2. *for every vertex v and color $c \in S(v)$, $\deg_S(v, c) \leq d$ (recall that $\deg_S(v, c)$ denotes the c-degree of v which is the number of neighbors u of v with color $c \in S(u)$).*

Then, there exists a proper coloring of G from these lists.

A weaker version of this result obtained by replacing $(1+\varepsilon)$ above with some absolute constant appeared earlier in [31] (see also [3, Proposition 5.5.3] and [20]).

3 Two New Palette Sparsification Theorems

We present our new palette sparsification theorems in Result 1 and Result 2 in this section. We postpone the proof of the optimality of Result 1 (the lower bound on sampled-list sizes) to the full version of the paper as it is a basic argument. Instead we give the more interesting proof of the optimality of Result 2 in almost full details in this section.

3.1 Palette Sparsification for $(1+\varepsilon)\Delta$ Coloring

We start with our improved palette sparsification theorem for $(1+\varepsilon)\Delta$ coloring.

▶ **Theorem 5.** *For every $\varepsilon \in (0, 1/2)$, there exists an integer $n(\varepsilon) \geq 1$ such that the following is true. Let $G(V,E)$ be any graph with $n \geq n(\varepsilon)$ vertices and maximum degree Δ, and define $C := C(\varepsilon) = (1+\varepsilon)\cdot\Delta$. Suppose for every vertex $v \in V$, we independently sample a set $L(v)$ of colors of size $\ell := \left(10\sqrt{\log n}/\varepsilon^{1.5}\right)$ uniformly at random from colors $\{1,\ldots,C\}$. Then, with high probability, there exists a proper coloring of G from lists $L(v)$ for every $v \in V$.*

We shall note that in contrast to Theorem 5, it was shown in [5] that for the more stringent problem of $(\Delta+1)$ coloring, sampling $\Omega(\log n)$ colors per vertex is necessary. As such, Theorem 5 presents a separation between these two problems in palette sparsification.

Proof of Theorem 5

The proof of this theorem is by showing that the lists sampled for vertices can be adjusted so that they satisfy the requirement of Proposition 4; we then apply this proposition to obtain a list-coloring of G from the sampled lists.

Recall that $\deg_L(v,c)$ is the c-degree of vertex v in lists L. For every $c \in L(v)$,

$$\mathbb{E}\left[\deg_L(v,c)\right] := \sum_{u \in N(v)} \mathbb{P}\left(u \text{ samples } c \text{ in } L(u)\right) \leq \Delta \cdot \frac{\ell}{C} = \frac{\ell}{1+\varepsilon}. \tag{1}$$

Now *if* $\deg_L(v,c)$ was concentrated enough so that $\max_{v,c} \deg_L(v,c) = (1-\Theta(\varepsilon))\cdot\ell$, we would have been done already: by Proposition 4, there is always a proper coloring of G from such lists (take the parameter d to be $\max_{v,c}\deg_L(v,c)$ and so size of each list is $(1+\Theta(\varepsilon))d$). Unfortunately however, it is easy to see that as $\ell = \Theta(\sqrt{\log n})$ in general no such concentration is guaranteed.

We fix the issue above by showing existence of a *subset* $\widehat{L}(v)$ of each list $L(v)$ such that these new lists can indeed be used in Proposition 4. The argument is intuitively as follows: the probability that $\deg_L(v,c)$ deviates significantly from its expectation is $2^{-\Theta(\ell)} = 2^{-\Theta(\sqrt{\log n})}$ by a simple Chernoff bound. Moreover, the probability that $\Omega(\sqrt{\log n})$ colors in $L(v)$ all deviate from their expectation can be bounded by $\left(2^{-\Theta(\sqrt{\log n})}\right)^{\Omega(\sqrt{\log n})}$ (ignoring dependency issues for the moment). This probability is now $n^{-\Theta(1)}$, enough for us to take a union bound over all vertices. As such, by removing some fraction of the colors from the list of each vertex, we can satisfy the c-degree requirements for applying Proposition 4 and conclude the proof. We now formalize this.

We say that a color $c \in L(v)$ is *bad* for v iff $\deg_L(v,c) > (1+\varepsilon/2)\cdot\mathbb{E}\left[\deg_L(v,c)\right]$. As the choice of color c for each vertex $u \in N(v)$ is independent, by Eq (1) and Chernoff bound (Proposition 28),

$$\mathbb{P}\left(\deg_L(v,c) > (1+\varepsilon/2)\cdot\mathbb{E}\left[\deg_L(v,c)\right]\right) \leq \exp\left(-\frac{\varepsilon^2}{12}\cdot\frac{\ell}{1+\varepsilon}\right). \tag{2}$$

Define $\mathsf{bad}(v)$ as the number of colors c in $L(v)$ that are bad for vertex v. We note that by the sampling process in Theorem 5, conditioning on some colors being bad for v can only reduce the chance of the remaining colors being bad for v. As such, by Eq (2),

$$\begin{aligned}\mathbb{P}\left(\mathsf{bad}(v) \geq \varepsilon/4 \cdot \ell\right) &\leq \binom{\ell}{\varepsilon/4 \cdot \ell} \cdot \exp\left(-\frac{\varepsilon^2}{12} \cdot \frac{\ell}{1+\varepsilon}\right)^{\varepsilon \cdot \ell/4} \\ &\leq 2^{\ell} \cdot \exp\left(-\frac{\varepsilon^3}{72} \cdot \ell^2\right) \leq \exp\left(-20 \log n\right).\end{aligned}$$

(by the choice of $\ell = 10\sqrt{\log n}/\varepsilon^{1.5}$ and as $\varepsilon < 1/2$ is sufficiently smaller than n)

By a union bound over all n vertices, with high probability, for every vertex v, $\mathsf{bad}(v) \leq \varepsilon \cdot \ell/4$. We let $\widehat{L}(v)$ to be a subset of $L(v)$ obtained by removing all bad colors from $L(v)$. For any $c \in \widehat{L}(v)$:

$$\deg_{\widehat{L}}(v,c) \leq \deg_{L}(v,c) \leq (1+\varepsilon/2) \cdot \frac{\ell}{1+\varepsilon} \leq (1-\varepsilon/3) \cdot \ell. \qquad \text{(for } \varepsilon < 1/2\text{)}$$

On the other hand, as $\mathsf{bad}(v) \leq \varepsilon \cdot \ell/4$, we have $\left|\widehat{L}(v)\right| \geq (1-\varepsilon/4) \cdot \ell$. As such, by Proposition 4 (as ε is a constant with respect to ℓ), we can list-color G from lists $\widehat{L}$ and consequently also L, finalizing the proof. ◀

3.2 Palette Sparsification for Triangle-Free Graphs

We now prove a palette sparsification theorem for triangle-free graphs.

▶ **Theorem 6.** *Let $G(V,E)$ be any n-vertex triangle-free graph with maximum degree Δ. Let $\gamma \in (0,1)$ be a parameter and define $C := C(\gamma) = \left(\frac{9\Delta}{\gamma \cdot \ln \Delta}\right)$. Suppose for every vertex $v \in V$, we independently sample a set $L(v)$ of size $b \cdot (\Delta^{\gamma} + \sqrt{\log n})$ uniformly at random from colors $\{1,\ldots,C\}$ for an appropriate absolute positive constant b. Then, with high probability there exists a proper coloring of G from lists $L(v)$ for every vertex $v \in V$.*

It is known that there are triangle-free graphs with chromatic number $\Omega(\frac{\Delta}{\ln \Delta})$ [10] (In fact this bound holds even for graphs with arbitrarily large girth not only girth > 3). Theorem 6 then shows that one can match the chromatic number of these graphs asymptotically by sampling a small number of colors per vertex (almost as small as $O(\Delta^{o(1)} + \sqrt{\log n})$).

Proof of Theorem 6

As we already saw in the proof of Theorem 5, looking at the sampled lists $L(v)$ of vertices as a list-coloring problem with constraints on c-degrees can be quite helpful in proving the corresponding palette sparsification result. We take the same approach in proving Theorem 6 as well. However, unlike for $(1+\varepsilon)\Delta$ coloring, to the best of our knowledge, no such list-coloring results (with constraints on c-degrees *instead* of maximum degree) are known for coloring triangle-free graphs. Our main task here is then exactly to prove such a result formalized as follows.

▶ **Proposition 7.** *There exists an absolute constant d_0 such that for all $d \geq d_0$ the following holds. Suppose $G(V,E)$ is a triangle-free graph with lists $S(v)$ for every $v \in V$ such that:*

1. *for every vertex v, $|S(v)| \geq 8 \cdot \frac{d}{\ln d}$, and*
2. *for every vertex v and color $c \in S(v)$, $\deg_S(v,c) \leq d$.*

Then, there exists a proper coloring of G from these lists.

We give the proof of Theorem 6 assuming Proposition 7 here. The proof of Proposition 7 itself is technical and detailed and thus even though interesting on its own, we opted to postpone it to the full version to preserve the flow of the paper here.

Proof of Theorem 6. We prove this theorem with the weaker bound of $O(\Delta^\gamma + \log n)$ (as opposed to $O(\Delta^\gamma + \sqrt{\log n})$) for the number of sampled colors. The extension to the improved bound with $O(\sqrt{\log n})$ dependence is exactly as in the proof of Theorem 5 and is thus omitted.

Let $\ell := (\Delta^\gamma + 100 \ln n)$ and suppose each vertex samples ℓ colors from $\{1, \ldots, C\}$ for $C := C(\gamma) = \left(\frac{9\Delta}{\gamma \cdot \ln \Delta}\right)$. Let $p := \ell / C$ which is equal to the probability that any vertex v samples a particular color in $L(v)$. We have,

$$\mathbb{E}\left[\deg_L(v, c)\right] = \sum_{u \in N(v)} \mathbb{P}\left(u \text{ samples } c \text{ in } L(u)\right) \leq p \cdot \Delta.$$

Note that as $p \cdot \Delta \geq p \cdot C = \ell \geq 100 \ln n$, a simple application of Chernoff bound plus union bound ensures that, for every vertex v and color c, $\deg_L(v, c) \leq (1.1) \cdot p\Delta$ with high probability. In the following, we condition on this event.

Let $d := (1.1) \cdot p\Delta$. By the above conditioning, c-degree of every vertex $v \in V$ is at most d. In order to apply Proposition 7 to graph G with lists L, we only need to prove that $\ell \geq \frac{8d}{\ln d}$. We prove that in fact $\ell \cdot \ln \ell \geq 8d$ which implies the desired bound as $\ell = p \cdot C \leq p \cdot \Delta \leq d$. We have,

$$\ell \cdot \ln \ell \geq (p \cdot C) \cdot \ln\left(\Delta^\gamma\right) = p \cdot \left(\frac{9\Delta}{\gamma \cdot \ln \Delta}\right) \cdot \gamma \cdot \ln \Delta = 9 \cdot p\Delta > 8d.$$

(as $\Delta^\gamma < \ell = p \cdot C$ and by the choice of C)

The proof now follows from applying Proposition 7 to lists L. ◀

Asymptotic Optimality of the Bounds in Theorem 6

We now prove the optimality of Theorem 6 up to constant factors.

▶ **Proposition 8.** *There exists a distribution on n-vertex graphs with maximum degree $\Delta = \Theta(n^{1/3})$ such that for every $\gamma < 1/16$ and $C := C(\gamma) = \frac{\Delta}{16\gamma \cdot \ln \Delta}$ the following is true. Suppose we sample a graph $G(V, E)$ from this distribution and then for each vertex $v \in V$, we independently pick a set $L(v)$ of colors with size Δ^γ uniformly at random from colors $\{1, \ldots, C\}$; then, with high probability there exists no proper coloring of G wherein the color of every vertex $v \in V$ sis chosen from $L(v)$.*

Let $\mathcal{G}_{n,p}$ denote the Erdős-Rényi distribution of random graphs on n vertices in which each edge is chosen independently with probability p. Define the following distribution $\mathcal{G}_{n,p}^{-K_3}$ on triangle-free graphs: Sample a graph G from $\mathcal{G}_{n,p}$, then remove *every* edge that was part of a triangle originally. Clearly, the graphs output by $\mathcal{G}_{n,p}^{-K_3}$ are triangle-free. Throughout this section, we take $p = \Theta(n^{-2/3})$ (the choice of the constant will be determined later).

We prove Proposition 8 by considering the distribution $\mathcal{G}_{n,p}^{-K_3}$. However, we first present some basic properties of distribution $\mathcal{G}_{n,p}$ needed for our purpose. The proofs are simple exercises in random graph theory and can be found in the full version of the paper. In the following, let $t(G)$ denote the number of triangles in G and $\alpha(G)$ denote the maximum independent set size, and recall that $\Delta(G)$ denotes the maximum degree of G.

▶ **Lemma 9.** *For $G \sim \mathcal{G}_{n,p}$, $\mathbb{E}\left[t(G)\right] \leq (np)^3$, and $t(G) \leq (1 + o(1))\,\mathbb{E}\left[t(G)\right]$ w.h.p.*

▶ **Lemma 10.** *For $G \sim \mathcal{G}_{n,p}$, $\mathbb{E}\left[\alpha(G)\right] \leq \frac{3 \cdot \ln(np)}{p}$, and $\alpha(G) \leq \frac{3 \cdot \ln(np)}{p}$ w.h.p.*

▶ **Lemma 11.** *For $G \sim \mathcal{G}_{n,p}$, $\Delta(G) \leq 2np$ w.h.p.*

Proof of Proposition 8. Let $p := \frac{1}{3} \cdot (n)^{-2/3}$ for this proof and consider the distribution $\mathcal{G}_{n,p}^{-K_3}$. Moreover, let $\mathcal{L}$ denote the distribution of lists of colors sampled for vertices. By Lemma 11, the maximum degree of $G \sim \mathcal{G}_{n,p}$ and consequently $G \sim \mathcal{G}_{n,p}^{-K_3}$ is at most $\widetilde{\Delta} := 2np$ with high probability. Throughout the following argument, we condition on this event. This can only change the probability calculations by a negligible factor (that we ignore for the simplicity of exposition). This way, the number of colors sampled in $\mathcal{L}$ can be assumed to be at most $C := \frac{\widetilde{\Delta}}{16\gamma \cdot \ln \widetilde{\Delta}}$. We further use $q := \widetilde{\Delta}^{\gamma}/C$ to denote the probability that a color c is sampled in list $L(v)$ of a vertex v.

For a graph $G(V, E) \sim \mathcal{G}_{n,p}^{-K_3}$ and lists $L \sim \mathcal{L}$, let $V_1, \ldots, V_C$ be a collection of subsets of V (not necessarily disjoint) where for every $c \in [C]$, V_c denotes the vertices v that sampled the color c in their list $L(v)$. As each color is sampled with probability q by a vertex, and the choices are independent across vertices, a simple application of Chernoff bound ensures that with high probability, $|V_c| \leq 2q \cdot n$ for all c. We also condition on this event in the following (and similarly as before ignore the negligible contribution of this conditioning to the probability calculations below).

Let δ denote the probability of "error" i.e., the event that the sampled colors do not lead to a proper coloring of the graph. An averaging argument implies that there exists a fixed set of lists $L \sim \mathcal{L}$ such that for G sampled from $\mathcal{G}_{n,p}^{-K_3}$, the error probability of L on G is at most δ. Fix such a choice of L in the following. We will show that $\delta = 1 - o(1)$.

Recall that $G \sim \mathcal{G}_{n,p}^{-K_3}$ is chosen independent of the lists L (by definition of palette sparsification). For any graph G, define:

- $\mu_L(G) := \max_{(U_1,\ldots,U_C)} \sum_{c=1}^{C} |U_c|$ where all U_c's are *disjoint*, each $U_c \subseteq V_c$, and $G[U_c]$ is an independent set.

As we have fixed the choice of the lists L, the function $\mu_L(\cdot)$ is fixed at this point and its value only depends on G. A necessary condition for G to be colorable from the lists L is that $\mu_L(G) = n$. This is because (i) any proper coloring of G from lists L necessarily induces an independent set inside each V_c; (ii) these independent sets are disjoint and hence we can take them as a feasible solution $(U_1, \ldots, U_C)$ to $\mu_L(G)$; (iii) these independent sets cover all vertices of G. Our task is now to bound the probability that $\mu(G) = n$ to lower bound δ.

Firstly, we can switch from the distribution $\mathcal{G}_{n,p}^{-K_3}$ to $\mathcal{G}_{n,p}$ using the following equation (recall that $t(G)$ denotes the number of triangles):

$$\mathop{\mathbb{E}}_{G \sim \mathcal{G}_{n,p}^{-K_3}} [\mu_L(G)] \leq \mathop{\mathbb{E}}_{H \sim \mathcal{G}_{n,p}} [\mu_L(H) + 3 \cdot t(H)]\,. \tag{3}$$

This is because any graph $G \sim \mathcal{G}_{n,p}^{-K_3}$ is obtained by removing edges of every triangle in a graph $H \sim \mathcal{G}_{n,p}$ and removing these edges can only increase the total size of a collection of *disjoint* independent sets (namely, the value of μ_L) by the number of vertices in the triangles (in fact, by at most two vertices from each triangle). We can upper bound the second-term in Eq (3) using Lemma 9. We now bound the first term. In the following, let $n_c := |V_c|$ for $c \in [C]$. We have,

$$\mathop{\mathbb{E}}_{H\sim\mathcal{G}_{n,p}}[\mu_L(H)] \leq \mathop{\mathbb{E}}_{H\sim\mathcal{G}_{n,p}}\left[\sum_{c=1}^{C}\alpha(H[V_c])\right],$$

(by removing the disjointness condition between sets U_c's we can only increase value of $\mu_L(H)$)

$$= \sum_{c=1}^{C}\mathop{\mathbb{E}}_{H_c\sim\mathcal{G}_{n_c,p}}[\alpha(H_c)]$$

(by linearity of expectation and as for every $c\in[C]$, $H[V_c]$ is sampled from $\mathcal{G}_{n_c,p}$)

$$\leq \sum_{c=1}^{C}\frac{3\cdot\ln(n_c p)}{p} \qquad \text{(by Lemma 10)}$$

$$\leq C\cdot\frac{3\cdot\ln(2qn\cdot p)}{p} \qquad \text{(as we conditioned on } n_c\leq 2q\cdot n\text{)}$$

$$= \frac{\widetilde{\Delta}}{16\gamma\cdot\ln\widetilde{\Delta}}\cdot\frac{3\cdot\ln(q\cdot\widetilde{\Delta})}{(\widetilde{\Delta}/2n)} \qquad \text{(by definitions of } C \text{ and } \widetilde{\Delta}\text{)}$$

$$= \frac{6n}{16}\cdot\frac{\ln(q\cdot\widetilde{\Delta})}{\ln(\widetilde{\Delta}^{\gamma})} \qquad \text{(by a simple re-arranging of terms)}$$

$$< \frac{6n}{8}. \qquad \text{(as } \ln\left(q\cdot\widetilde{\Delta}\right)=\ln\left(\widetilde{\Delta}^{\gamma}\cdot 16\gamma\cdot\ln\widetilde{\Delta}\right)<2\ln\left(\widetilde{\Delta}^{\gamma}\right)\text{)}$$

Plugging this in Eq (3) together with Lemma 9 to bound the second term, implies that:

$$\mathop{\mathbb{E}}_{G\sim\mathcal{G}_{n,p}^{-K_3}}[\mu_L(G)] \leq \frac{6n}{8}+3\cdot(\frac{n^{1/3}}{3})^3 < \frac{7n}{8}.$$

Finally, by Lemmas 9 and 10, $\mu_L(G)<n$ w.h.p. This implies that $\delta=1-o(1)$ as needed. ◀

4 A Local Version of Palette Sparsification

We now give a "local version" (see, e.g. [14, 11]) of the palette sparsification theorem.

▶ **Theorem 12.** *Let $G(V,E)$ be any n-vertex graph and assume each vertex $v\in V$ is given a list $S(v)$ of colors. Suppose for every vertex $v\in V$, we independently sample a set $L(v)$ of colors of size ℓ uniformly at random from colors in $S(v)$. Then,*
1. *if $S(v)$ is* any arbitrary set *of $(1+\varepsilon)\cdot\deg(v)$ colors and $\ell=\Theta(\varepsilon^{-1}\cdot\log n)$ for $\varepsilon>0$,*
2. *or if $S(v)=\{1,\ldots,\deg(v)+1\}$ and $\ell=\Theta(\log n)$,*

then, with high probability, there exists a proper coloring of G from lists $L(v)$ for $v\in V$.

The main part of the proof of Theorem 12 is Part 2 as the proof of the first part follows almost directly from this proof. However, we start with a standalone proof of Part 1 for intuition and then sketch the proof of Part 2, which involves the bulk of our effort.

Proof of Theorem 12 – Part 1. Fix any $\varepsilon>0$ and suppose we sample $\ell:=\frac{10}{\varepsilon}\cdot\ln n$ colors $L(v)$ from $S(v)$ for every vertex $v\in V$. Consider the following process:

1. Iterate over vertices v in an *arbitrary* order and for each vertex v, let $N^{<}(v)$ denote the neighbors of v that appear before v in this ordering.
2. For each vertex v, if there exists a color $c(v)$ in $L(v)$ that is not used to color any vertex $u\in N^{<}(v)$, color v with $c(v)$. Otherwise **abort**.

We argue that this procedure will terminate with high probability without having to **abort**. This ensures that G is colorable from sampled lists L, thus proving Part 1 of Theorem 12.

$$\begin{aligned}\mathbb{P}(\textbf{abort}) &\leq \sum_v \mathbb{P}\left(L(v) \text{ is a subset of colors chosen for } N^{<}(v)\right) && \text{(by union bound)}\\ &\leq \sum_v \left(\frac{|N^{<}(v)|}{|S(v)|}\right)^{\ell} \leq n \cdot \left(\frac{\deg(v)}{(1+\varepsilon)\cdot\deg(v)}\right)^{\ell} \leq n\cdot(1-\varepsilon/2)^{\ell} \\ &&& \text{(by the sampling without replacement procedure of Theorem 12)}\\ &\leq n\cdot\exp\left(-\frac{\varepsilon}{2}\cdot\frac{10}{\varepsilon}\cdot\ln n\right) = n^{-4}. && \text{(by the choice of } \ell)\end{aligned}$$

This concludes the proof of Part 1 of Theorem 12. ◀

Proof Sketch of Theorem 12 – Part 2. In order to prove the second part of Theorem 12, we follow the approach of [5] for $(\Delta+1)$ coloring problem. The key difference is that the graph decomposition of the graph into *sparse* and *dense* parts that played a key role in [5] is no longer applicable to $(\deg+1)$ coloring. In the following, we first give a new graph decomposition tailored to $(\deg+1)$ coloring and states its main properties as well as its differences with similar decompositions for $(\Delta+1)$ coloring in [18, 13, 5] (themselves based on [30]). The next step is then to show that this decomposition, even though "weaker" than the one for $(\Delta+1)$ coloring, still has enough structure to carry out the proof for $(\deg+1)$ coloring along the lines of the one for $(\Delta+1)$ coloring in [5] with the main difference being on how we handle the "sparse" vertices.

A Graph Decomposition for $(\deg+1)$ Coloring. Let $\varepsilon \in (0,1)$ be a parameter. We define the following structures for any graph $G(V,E)$.

▶ **Definition 13.** *We say that an induced subgraph K of G is an **ε-almost-clique** iff:*
1. *For every $v \in K$, $\deg_G(v) \geq (1-8\varepsilon)\cdot\Delta(K)$ where we define $\Delta(K) := \max_{v\in K}\deg_G(v)$;*
2. *$(1-\varepsilon)\cdot\Delta(K) \leq |V(K)| \leq (1+8\varepsilon)\cdot\Delta(K)$;*
3. *Any vertex $v \in K$ has at most $8\varepsilon\cdot\Delta(K)$* non-neighbors *(in G)* inside *K;*
4. *Any vertex $v \in K$ has at most $9\varepsilon\cdot\Delta(K)$* neighbors *(in G)* outside *K.*

Definition 13 can be seen as a natural analogue of (Δ,ε)-almost-cliques defined in [5]. The main difference is that instead of having dependence on the global parameter Δ in a (Δ,ε)-almost-clique of [5], our ε-almost-cliques only depend on $\Delta(K)$ which is a $(1+\Theta(\varepsilon))$-approximation of the degree of every vertex in K (thus can be much smaller than Δ).

▶ **Definition 14.** *We say a vertex $v \in G$ is **ε-sparse** iff there are at least $\varepsilon^2\cdot\binom{\deg(v)}{2}$* non-edges *in the neighborhood of v.*

Again, Definition 14 is a natural analogue of sparse vertices in [5, 18, 13] by replacing the dependence on Δ with $\deg(v)$ instead.

▶ **Definition 15.** *We say a vertex $v \in G$ is **ε-uneven** iff for* at least $\varepsilon\cdot\deg(v)$ *neighbors u of v, we have* $\deg(v) < (1-\varepsilon)\cdot\deg(u)$.

Roughly speaking, a vertex v is considered uneven if it has a "sufficiently large" number of neighbors with "sufficiently larger" degree than v. Definition 15 is tailored specifically to $(\deg+1)$ coloring problem and does not have an analogue in [5, 18, 13] for $(\Delta+1)$ coloring. We prove the following decomposition result using the definitions above.

▶ **Lemma 16** (Graph Decomposition for $(\deg+1)$ Coloring). *For any sufficiently small $\varepsilon > 0$, any graph $G(V, E)$ can be partitioned into $V := V^{\mathsf{uneven}} \sqcup V^{\mathsf{sparse}} \sqcup K_1 \sqcup \ldots \sqcup K_k$ such that:*

1. *For every $i \in [k]$, the induced subgraph $G[K_i]$ is an ε-almost-clique;*
2. *Every vertex in V^{sparse} is $(\varepsilon/2)$-sparse;*
3. *Every vertex in V^{uneven} is $(\varepsilon/4)$-uneven.*

The key difference of Lemma 16 with prior decompositions for $(\Delta+1)$ coloring in [30, 5, 18, 13] is the introduction of V^{uneven} that captures vertices with "sufficiently large" higher degree neighbors. Allowing for such vertices is (seemingly) crucial for this type of decomposition that depends on the local degrees of vertices as opposed to maximum degree[3]. We postpone the proof of this lemma to the full version of the paper.

Coloring the Graph Using the Decomposition. For the rest of the proof, fix a decomposition of the graph $G(V, E)$ with some sufficiently small **absolute constant** $\varepsilon > 0$ (taking $\varepsilon = 10^{-4}$ would certainly suffice). In the following, we show that we can handle both V^{uneven} and V^{sparse} vertices first, and then color the almost-cliques using a result of [5] almost in a black-box way. As such, the main difference between our work and [5] (beside the decomposition) is in the treatment of vertices in $V^{\mathsf{uneven}} \cup V^{\mathsf{sparse}}$.

As in [5], the proof consists of two parts. We first show that $V^{\mathsf{uneven}} \cup V^{\mathsf{sparse}}$ can be colored from the sampled lists, and then show how to color each almost-clique given any arbitrary coloring of vertices outside the almost-clique. Before we move on, we make an assumption (without loss of generality) that is used in our concentration bounds.

▶ **Assumption 17.** *We may and will assume that degree of every vertex is at least $D_{\mathsf{min}} := \alpha \cdot \varepsilon^{10} \cdot \log n$ for some sufficiently large **absolute constant** $\alpha > 0$. This is without loss of generality as by sampling $\Theta(\log n)$ colors, any vertex with lower degree will have $L(v) = S(v)$ and hence we can greedily color these vertices after finding a coloring of the rest of the graph.*

Step one. The main part of the argument is the following lemma.

▶ **Lemma 18.** *Suppose for every vertex $v \in V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}$, we sample a set $L(v)$ of $\Theta(\varepsilon^{-6} \cdot \log n)$ colors independently and uniformly at random from $S(v) := \{1, \ldots, \deg(v) + 1\}$. Then, with high probability, the induced subgraph $G[V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}]$ can be properly colored from the sampled lists.*

We construct the coloring of Lemma 18 in two steps. The first step is to create "excess" colors on vertices, reducing the problem essentially to $(1 + \varepsilon)\deg$ coloring, and then using the simple argument for the proof of Part 1 of Theorem 12) to finalize this case as well. One important bit is that the first step of this argument should be done *simultaneously* for both V^{uneven} and V^{sparse}. We now sketch some the key ideas of the proof.

For the proof of Lemma 18, we need to partition vertices in V^{sparse} and V^{uneven} further in order to be able to handle the disparity in degree of vertices. As such, we define:

- $\psi := \varepsilon^2/32$: a parameter used throughout the definitions in this part for ease of notation.
- V^{small}: Let $\text{Small}(v) := \{u \in N(V) : \deg(u) < d_{\mathsf{small}}(v)\}$ where $d_{\mathsf{small}}(v) := \psi \cdot \deg(v)$. We define $V^{\mathsf{small}} \subseteq V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}$ as all vertices v with $|\text{Small}(v)| \geq 2d_{\mathsf{small}}(v)$.
- V^{large}: Let $\text{Large}(v) := \{u \in N(v) : \deg(u) > d_{\mathsf{large}}(v)\}$ where $d_{\mathsf{large}}(v) := 2\deg(v)$. We define $V^{\mathsf{large}} \subseteq V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}$ as all vertices v with $|\text{Large}(v)| \geq \psi \cdot \deg(v)$.[4]

[3] For instance, consider a vertex of degree d that is incident to d vertices of a $2d$-clique. Such a vertex is neither sparse (its neighborhood is a clique), nor belongs to an almost-clique for small $\varepsilon < 1$.

[4] We remark that the change in the place where ψ used in the two definitions above is intentional and not a typo.

As stated earlier, the goal of our first step is to construct excess colors for vertices. As it will become evident shortly, vertices in V^{small} actually do not need require having excess colors to begin with (roughly speaking, after coloring their very "low degree" neighbors in $\text{Small}(v)$, we are anyway left with many excess colors). Hence, we ignore these vertices in the first step altogether and handle them directly in the second one. Another important remark about the first step is that even though its goal is to color only $V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}$ (minus V^{small}), we assume *all* vertices of the graph (including almost-cliques) participate in its coloring procedure. This is only to simplify the math and after this step we simply *uncolor* all vertices that are not in $V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}$.

Creating Excess Colors. We start with the following coloring procedure as our first step:

Algorithm 1 FirstStepColoring: A procedure for finding a (partial) coloring of $G[V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}]$.

1. Iterate over vertices of V in an arbitrary order.
2. For every vertex v, **activiate** v w.p. $p_{\mathtt{active}} := \psi/16 \ \ (= \Theta(\varepsilon^2))$.
3. For every activated vertex v, pick a color $c_1(v)$ uniformly at random from $L(v)$ and if $c(v)$ is not used to color any neighbor of v so far, **color** v with $c_1(v)$.

We shall note right away that distribution of $c_1(v)$ for every vertex v in FirstStepColoring is simply uniform over $S(v)$. For any vertex $v \in V$, let $S_1(v)$ denote the list of available colors $S(v)$ after removing the colors assigned to neighbors of v in this procedure. Similarly, let $\deg_1(v)$ denote the degree of v after removing the colored neighbors of v. We show that $S_1(v)$ is "sufficiently larger" than $\deg_1(v)$ for all vertices in $V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}} \setminus V^{\mathsf{small}}$. Formally,

▶ **Lemma 19.** *There exists an absolute constant $\alpha \in (0,1)$ such that with high probability, for every $v \in V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}} \setminus V^{\mathsf{small}}$, we have $|S_1(v)| \geq \deg_1(v) + \alpha \cdot \varepsilon^6 \cdot \deg(v)$.*

The proof of of this lemma is given in three parts, each for coloring one of the sets V^{uneven}, V^{large} and $V^{\mathsf{sparse}} \setminus (V^{\mathsf{small}} \cup V^{\mathsf{large}})$ separately. The first two have an almost identical proof and are based on a novel argument – the third part uses a different argument which on a high level is similar to the approach of [5] (and [15, 18, 13], all rooted in an earlier work of [25]) for coloring sparse vertices (according to a global definition of sparse based on Δ), although several new challenges has to be addressed there as well.

▶ **Lemma 20.** *W.h.p. for all $v \in V^{\mathsf{uneven}}$: $|S_1(v)| \geq \deg_1(v) + \alpha \cdot \varepsilon^4 \cdot \deg(v)$.*

Proof. Let $\theta := (\varepsilon/4)$ and recall that all vertices in V^{uneven} are θ-uneven by Lemma 16. Fix a vertex v in V^{uneven} and let $U(v)$ be the neighbors u of v where $\deg(v) < (1-\theta) \cdot \deg(u)$. As v is θ-uneven $|U(v)| \geq \theta \cdot \deg(v)$. For any $u \in U(v)$, let $S_{\mathsf{ext}}(u) = S(u) \setminus S(v)$ denote the set of colors that are available (originally) to u but not to v. For $s_{\mathsf{ext}}(u) := |S_{\mathsf{ext}}(u)|$, we have,

$$s_{\mathsf{ext}}(u) = \deg(u) - \deg(v) \geq \deg(u) - (1-\theta) \cdot \deg(u) = \theta \cdot \deg(u). \tag{4}$$

We say that a vertex $u \in U(v)$ is *good* iff u is colored from $S_{\mathsf{ext}}(u)$ by FirstStepColoring. Let $n_{\mathsf{good}}(v)$ denote the number of good neighbors of v. It is easy to see that $|S_1(v)| \geq \deg_1(v) + n_{\mathsf{good}}(v)$ as colors of good vertices are not removed from $S(v)$. Our goal is to lower bound $n_{\mathsf{good}}(v)$ then. Define the following two events:

- $\mathcal{E}_{\mathsf{active}}$: For every vertex $u \in V$, the number of active neighbors of u, denoted by $\deg_{\mathsf{active}}(u)$, is between $(p_{\mathtt{active}}/2) \cdot \deg(u)$ and $(2p_{\mathtt{active}}) \cdot \deg(u)$.
- $\mathcal{E}^U_{\mathsf{active}}(v)$: The set $U^{\mathsf{active}}(v)$ of *active* vertices in $U(v)$ has size at least $(p_{\mathtt{active}}/2) \cdot \theta \cdot \deg(v)$.

By our Assumption 17 and a simple application of Chernoff bound, both event $\mathcal{E}_{\mathsf{active}}$ and $\mathcal{E}^{U}_{\mathsf{active}}(v)$ hold with high probability (recall the lower bound on size of $U(v)$) above. Note that both these events are only a function of the probability of activating each vertex and independent of choice of lists L. Hence, in the following we condition on these events (and all coins tosses for activation probabilities) and only consider the randomness with respect to choices in L.

Let $u_1, \ldots, u_k$ for $k := (p_{\mathtt{active}}/2) \cdot \theta \cdot \deg(v)$ be the first k vertices in $U^{\mathsf{active}}(v)$ according to the ordering of FirstStepColoring. Let $\mathcal{R}(u_i)$ denote all the random choices that govern whether u_i will be good or not. Note that by the time we process u_i at most $\deg_{\mathsf{active}}(u_i)$ colors from $S(u_i)$ may have been assigned to neighbors of u_i. Even if all of these colors are adversarially chosen to be in $S_{\mathsf{ext}}(u_i)$, the number of colors that if chosen by u_i make u_i a good vertex is at least:

$$s_{\mathsf{ext}}(u_i) - \deg_{\mathsf{active}}(u_i) \geq \theta \cdot \deg(u_i) - (2p_{\mathtt{active}}) \cdot \deg(u_i) > (\theta/2) \cdot \deg(u_i).$$
(by Eq (4) and event $\mathcal{E}_{\mathsf{active}}$, respectively and since $p_{\mathtt{active}} = \Theta(\varepsilon^2) < \theta/4$)

Even conditioned on everything else, this choice is only a function of $c_1(u_i)$ chosen uniformly at random from $S(u_i)$. As such,

$$\mathbb{P}\left(u_i \text{ is good} \mid \mathcal{R}(u_1), \ldots, \mathcal{R}(u_{i-1})\right) \geq \frac{(\theta/2) \cdot \deg(u_i)}{\deg(u_i)+1} \geq (\theta/3).$$

This implies that (i) $\mathbb{E}\left[n_{\mathsf{good}}(v)\right] \geq (\theta/3) \cdot k$ and (ii) the distribution of good vertices among first k vertices in $U^{\mathsf{active}}(v)$ stochastically dominates the binomial distribution $\mathcal{B}(k, \theta/3)$. By a basic concentration of binomial distributions (say by using Chernoff bound in Proposition 28):

$$\mathbb{P}\left(n_{\mathsf{good}}(v) < (\theta/6) \cdot k\right) \leq \exp\left(-\Theta(1) \cdot \theta \cdot k\right) = \exp\left(-\Theta(1) \cdot \varepsilon^4 \cdot \log n\right) \ll n^{-10}.$$
(by the choice of $\theta = \Theta(\varepsilon)$, $p_{\mathtt{active}} = \Theta(\varepsilon^2)$, k, and Assumption 17)

As $k = \Theta(\varepsilon^3 \cdot \deg(v))$ and $\theta = \Theta(\varepsilon)$, we obtain that w.h.p. $n_{\mathsf{good}}(v) \geq \Theta(\varepsilon^4) \cdot \deg(v)$. ◀

The following lemma has a similar proof as Lemma 20 and is postponed to the full version.

▶ **Lemma 21.** *W.h.p. for all $v \in V^{\mathsf{large}}$: $|S_1(v)| \geq \deg_1(v) + \alpha \cdot \varepsilon^4 \cdot \deg(v)$.*

Finally, the following lemma also has a relatively standard proof by-now and is postponed to the full version.

▶ **Lemma 22.** *Wh.p. for all $v \in V^{\mathsf{sparse}} \setminus (V^{\mathsf{small}} \cup V^{\mathsf{large}})$: $|S_1(v)| \geq \deg_1(v) + \alpha \cdot \varepsilon^6 \cdot \deg(v)$.*

Lemma 19 now follows directly from Lemmas 20, Lemma 21 and 22 and a union bound.

Exploiting Excess Colors. For the second step, consider the following procedure:

Algorithm 2 SecondStepColoring: A procedure for finishing the proper coloring of $G[V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}]$.

1. Iterate over uncolored vertices $v \in V^{\mathsf{sparse}} \cup V^{\mathsf{uneven}}$ in an *arbitrary* order and for each vertex v, let $N^{<}(v)$ denote the neighbors of v that appear before v in this ordering *plus* all neighbors of v that have been colored in the first step.
2. For each vertex v, if there exists a color in $L(v)$ that is not used to color any vertex $u \in N^{<}(v)$, color v with this color. Otherwise **abort**.

It is immediate that if SecondStepColoring does not **abort**, we find a proper coloring using the sampled colors in lists L. We can prove that **abort** happens with a small probability (the proof is postponed to the appendix).

▶ **Lemma 23.** *W.h.p.* SecondStepColoring *does* not **abort**.

Lemma 18 now follows from Lemmas 19 and 23 and a union bound.

Step two. In the second part of the proof, we are left with the coloring of almost-cliques from the sampled lists *after* fixing the colors of remaining vertices. This is done by the following lemma. This lemma is a simple generalization of a result of [5] for $(\Delta+1)$ coloring and the proof is via a simple "reduction" to the proof of the original lemma of [5].

Recall the definition of an ε-almost-cliques K in Definition 13. For a vertex $v \in K$, we define out-deg(v) as the number of neighbors of v that are outside K. Note that by definition of ε-almost-cliques, out-deg$(v) \leq 9\varepsilon \cdot \Delta(K)$. We prove the following lemma in the full version.

▶ **Lemma 24.** *Let K be an ε-almost-clique in G according to Definition 13 for some sufficiently small $\varepsilon > 0$ and define $\Delta(K) := \max_{v \in K} \deg(v)$. Suppose for every vertex $v \in K$, we* adversarially *pick a set $\overline{S}(v)$ of size at most* out-deg$(v) \leq 9\varepsilon \cdot \Delta(K)$ *from colors $\{1, \ldots, \deg(v)+1\}$. If for every vertex $v \in V$, we sample a set $L(v)$ of $\Theta(\varepsilon^{-1} \cdot \log n)$ colors independently from the set of colors $\{1, \ldots, \deg(v)+1\}$, then, with high probability, the induced subgraph $G[K]$ can be properly colored from the lists $L(v) \setminus \overline{S}(v)$ for $v \in C$.*

Proof of Theorem 12 – Part 2. We fix a decomposition of the graph G according to Lemma 16 for some sufficiently small absolute constant $\varepsilon > 0$ (taking $\varepsilon = 10^{-4}$ would certainly suffice). Lemma 18 allows us to argue that with high probability, all vertices except for almost-cliques in the decomposition can be properly colored using the sampled lists. We fix such a coloring of those vertices. We then iterate over almost-cliques one by one, and invoke Lemma 24 to each almost-clique K_i by letting $\overline{S}(v)$ for every $v \in K_i$ to be the set of colors used so far in this process for coloring neighbors of v outside this almost-clique. This allows us to color this almost-clique in a way that its coloring can be extended to the partial coloring computed so far (with high probability). Iterating over all almost-cliques this way and using a union bound finalizes the proof. ◀

5 Sublinear Algorithms from Palette Sparsification

In this section, we describe some applications of our palette sparsification theorems to sublinear algorithms following the work of [5]. In the following, we give the definition of each of the two models of streaming algorithms and sublinear-time algorithms formally, followed by the resulting algorithms from palette sparsification for each one separately.

Streaming Algorithms. In the streaming model, edges of the graph are presented one by one to an algorithm that can make one or a few passes over the input and use a limited memory to process the stream and has to output the answer at the end of the last pass. In this paper, we only consider *single-pass* streaming algorithms. We can obtain the following algorithms from Results 1, 2, and 3.

▶ **Corollary 25.** *There exists randomized single-pass streaming algorithms for finding each of the following colorings with high probability:*

- *a $(1+\varepsilon)\Delta$ coloring of any general graph with $O_\varepsilon(n \log n)$ space;*
- *an $O(\frac{\Delta}{\gamma \cdot \ln \Delta})$ coloring of any triangle-free graph with $\widetilde{O}(n \cdot \Delta^{2\gamma})$ space;*
- *a $(1+\varepsilon)\deg$-list coloring of any general graph with $O_\varepsilon(n \cdot \log^2 n)$ space;*
- *a $(\deg+1)$ coloring of any general graph with $O(n \cdot \log^2 n)$ space.*

The streaming algorithms in Corollary 25 are basically as follows: we sample the colors in L at the beginning of the stream and throughout the stream whenever an edge (u, v) is presented, we check whether $L(u) \cap L(v) = \emptyset$ or not; if not we store this edge explicitly. At this point, obtaining the first two algorithms in Corollary 25 from Results 1 and 2 is straightforward (see also [5]). However, the results for the latter two parts does not immediately follow from the argument for other two (or the one in [5]). This is due to the fact that both $(1 + \varepsilon)\deg$ and $(\deg +1)$ problems are "local" problems with dependence on deg instead of Δ. We postpone the proof of this corollary to the full version of the paper. We conclude this part by noting that our results can be extended to dynamic streams where edges can be both inserted to and deleted from the stream by increasing the space of the algorithm with polylog(n) factors as was done in [5].

Sublinear-Time Algorithms. When designing sublinear-time algorithms, it is crucial to specify the data model as the algorithm cannot even read the entire input once. We assume the standard query model for sublinear-time algorithms on general graphs (see, e.g., [17, Chapter 10]). In this model, we have the following three types of queries (i) what is the degree of a vertex v; (ii) what is the i-th neighbor of a given vertex v; and (iii) whether a given pair of vertices (u, v) are neighbor to each other or not. We say an algorithm is *non-adaptive* if it asks all its queries in parallel in one go.

We can obtain the following algorithms from Results 1, 2, and 3.

▶ **Corollary 26.** *There exists randomized non-adaptive sublinear-time algorithms for finding each of the following colorings with high probability:*

- *a* $(1 + \varepsilon)\Delta$ *coloring of any general graph in* $\widetilde{O}_\varepsilon(n^{3/2})$ *time;*
- *an* $O(\frac{\Delta}{\gamma \cdot \ln \Delta})$ *coloring of any triangle-free graph in* $\widetilde{O}(n^{3/2+2\gamma})$ *time;*
- *a* $(1 + \varepsilon)\deg$*-list coloring of any general graph in* $\widetilde{O}(n^{3/2})$ *time;*
- *a* $(\deg +1)$ *coloring of any general graph in* $\widetilde{O}(n^{3/2})$ *time.*

The sublinear-time algorithms in Corollary 26 are again based on finding the edges of the conflict-graph E_{conflict} using $\widetilde{O}(\min\{n\Delta, n^2/\Delta\})$ queries for the case of $(1+\varepsilon)\Delta$ coloring and $\widetilde{O}(\min\{n\Delta, n^2/\Delta^{1-2\gamma}\})$ queries for triangle-free graphs. This can be done using the simple approach of [5] but as before that does not work for the last two parts. As such, in the full version of the paper, we give another simple way for finding edges of the conflict-graph using a small number of queries, and conclude the proof of Corollary 26.

References

1. Noga Alon and Sepehr Assadi. Palette sparsification beyond $(\Delta+1)$ vertex coloring. *CoRR*, abs/2006.10456, 2020. `arXiv:2006.10456`.
2. Noga Alon, Michael Krivelevich, and Benny Sudakov. Coloring graphs with sparse neighborhoods. *J. Comb. Theory, Ser. B*, 77(1):73–82, 1999.
3. Noga Alon and Joel Spencer. *The Probabilistic Method*. Fourth Edition, Wiley, 2016.
4. Amihood Amir, Oren Kapah, Tsvi Kopelowitz, Moni Naor, and Ely Porat. The family holiday gathering problem or fair and periodic scheduling of independent sets. In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 367–375, 2016.
5. Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 767–786, 2019.

6 Nikhil Bansal, Anupam Gupta, and Guru Guruganesh. On the Lovász theta function for independent sets in sparse graphs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 193–200, 2015.

7 Suman K. Bera, Amit Chakrabarti, and Prantar Ghosh. Graph coloring via degeneracy in streaming and other space-conscious models. *CoRR*, abs/1905.00566. To appear in ICALP 2020, 2019.

8 Anton Bernshteyn. The Johansson-Molloy theorem for DP-coloring. *Random Structures & Algorithms*, 54(4):653–664, 2019.

9 Andreas Björklund, Thore Husfeldt, and Mikko Koivisto. Set partitioning via inclusion-exclusion. *SIAM J. Comput.*, 39(2):546–563, 2009.

10 Béla Bollobás. Chromatic number, girth and maximal degree. *Discrete Mathematics*, 24(3):311–314, 1978.

11 Marthe Bonamy, Tom Kelly, Peter Nelson, and Luke Postle. Bounding χ by a fraction of Δ for graphs without large cliques. *arXiv preprint*, 2018. `arXiv:1803.01051`.

12 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of $(\Delta+1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 471–480, 2019.

13 Yi-Jun Chang, Wenzheng Li, and Seth Pettie. An optimal distributed $(\Delta + 1)$-coloring algorithm? In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 445–456, 2018.

14 Ewan Davies, Rémi de Joannis de Verclos, Ross J. Kang, and François Pirot. Colouring triangle-free graphs with local list sizes. *CoRR*, abs/1812.01534, 2018.

15 Michael Elkin, Seth Pettie, and Hsin-Hao Su. $(2\Delta - 1)$-edge-coloring is much easier than maximal matching in the distributed setting. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 355–370, 2015.

16 Paul Erdős, Arthur L Rubin, and Herbert Taylor. Choosability in graphs. In *Proc. West Coast Conf. on Combinatorics, Graph Theory and Computing, Congressus Numerantium*, volume 26, pages 125–157, 1979.

17 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

18 David G Harris, Johannes Schneider, and Hsin-Hao Su. Distributed $(\Delta + 1)$-coloring in sublogarithmic rounds. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 465–478. ACM, 2016.

19 Nicholas J. A. Harvey, Christopher Liaw, and Paul Liu. Greedy and local ratio algorithms in the mapreduce model. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, July 16-18, 2018*, pages 43–52, 2018.

20 Penny E Haxell. A note on vertex list colouring. *Combinatorics, Probability and Computing*, 10(4):345–347, 2001.

21 Anders Johansson. Asymptotic choice number for triangle free graphs. Technical report, unpublished manuscript, 1996.

22 Anders Johansson. The choice number of sparse graphs. Technical report, unpublished manuscript, 1996.

23 Jeong Han Kim. On Brooks' theorem for sparse graphs. *Combinatorics, Probability & Computing*, 4:97–132, 1995.

24 Michael Molloy. The list chromatic number of graphs with small clique number. *J. Comb. Theory, Ser. B*, 134:264–284, 2019.

25 Michael Molloy and Bruce A. Reed. A bound on the strong chromatic index of a graph. *J. Comb. Theory, Ser. B*, 69(2):103–109, 1997.

26 Michael Molloy and Bruce A. Reed. *Graph coloring and the probabilistic method*. New York I Springer, 2002.

27 Merav Parter. $(\Delta+1)$ coloring in the congested clique model. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 160:1–160:14, 2018.

28 Merav Parter and Hsin-Hao Su. Randomized $(\Delta + 1)$-coloring in $O(\log^* \Delta)$ congested clique rounds. In *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, pages 39:1–39:18, 2018.

29 Seth Pettie and Hsin-Hao Su. Distributed coloring algorithms for triangle-free graphs. *Inf. Comput.*, 243:263–280, 2015.

30 Bruce A. Reed. ω, Δ, and χ. *Journal of Graph Theory*, 27(4):177–212, 1998.

31 Bruce A. Reed. The list colouring constants. *Journal of Graph Theory*, 31(2):149–153, 1999.

32 Bruce A. Reed and Benny Sudakov. Asymptotically the list colouring constants are 1. *J. Comb. Theory, Ser. B*, 86(1):27–37, 2002.

33 Vojtech Rödl. On a packing and covering problem. *Eur. J. Comb.*, 6(1):69–78, 1985.

A Probabilistic Tools

We use the following standard probabilistic tools.

▶ **Proposition 27** (Lovász Local Lemma – symmetric form; cf. [3]). *Let $\mathcal{E}_1, \ldots, \mathcal{E}_n$ be n events such that each event $\mathcal{E}_i$ is mutually independent of all other events besides at most d, and $\mathbb{P}(\mathcal{E}_i) \leq p$ for all $i \in [n]$. If $e \cdot p \cdot (d+1) \leq 1$ (where $e = 2.71...$), then $\mathbb{P}\left(\wedge_{i=1}^{n} \overline{\mathcal{E}_i}\right) > 0$.*

▶ **Proposition 28** (Chernoff-Hoeffding bound; cf. [3]). *Let $X_1, \ldots, X_n$ be n independent random variables where each $X_i \in [0, b]$. Define $X := \sum_{i=1}^{n} X_i$. Then, for any $t > 0$,*

$$\mathbb{P}\left(|X - \mathbb{E}[X]| > t\right) \leq 2 \cdot \exp\left(-\frac{2t^2}{n \cdot b^2}\right).$$

Moreover, for any $\delta \in (0, 1)$,

$$\mathbb{P}\left(|X - \mathbb{E}[X]| > \delta \cdot \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{\delta^2 \cdot \mathbb{E}[X]}{3b}\right).$$

A function $f(x_1, \ldots, x_n)$ is called *c-Lipschitz* iff changing any single x_i can affect the value of f by at most c. Additionally, f is called *r-certifiable* iff whenever $f(x_1, \ldots, x_n) \geq s$, there exists at most $r \cdot s$ variables $x_{i_1}, \ldots, x_{i_{r \cdot s}}$ so that knowing the values of these variables certifies $f \geq s$.

▶ **Proposition 29** (Talagrand's inequality; cf. [26]). *Let $X_1, \ldots, X_n$ be n independent random variables and $f(X_1, \ldots, X_n)$ be a c-Lipschitz function; then for any $t \geq 1$,*

$$\mathbb{P}\left(|f - \mathbb{E}[f]| > t\right) \leq 2 \exp\left(-\frac{t^2}{2c^2 \cdot n}\right).$$

Moreover, if f is additionally r-certifiable, then for any $b \geq 1$,

$$\mathbb{P}\left(|f - \mathbb{E}[f]| > b + 30c\sqrt{r \cdot \mathbb{E}[f]}\right) \leq 4 \exp\left(-\frac{b^2}{8c^2 r\, \mathbb{E}[f]}\right).$$

B Sublinear Algorithms from Graph Partitioning

In this appendix, we deviate from our theme of palette sparsification and consider another technique for designing sublinear algorithms for graph coloring. A simple technique that lies at the core of various algorithms for graph coloring in different models is *random graph partitioning* (see, e.g. [27, 28, 19, 12, 7]). While the exact implementation of this technique varies significantly from one application to another, the basic idea is as follows: Partition the vertices of the graph G randomly into multiple parts $V_1, \ldots, V_k$, then color the induced subgraphs $G[V_1], \ldots, G[V_k]$ separately using disjoint palettes of colors for each subgraph. The hope is that each subgraph $G[V_i]$ has become "simpler enough" so that it can be colored "easily" with a "small" palette of colors.

We apply the same basic idea in this section. To state our result, we need some definitions first. We say that a family $\mathcal{G}$ of graphs is *hereditary* iff for every $G \in \mathcal{G}$, every induced subgraph of G also belongs to $\mathcal{G}$, namely, $\mathcal{G}$ is closed under vertex deletions.

▶ **Definition 30.** *Let $\mathcal{G}$ be a hereditary family of graphs and $\zeta : \mathbb{N}^+ \to \mathbb{N}^+$ be a non-decreasing function. We say that $\mathcal{G}$ is **ζ-colorable** iff every graph G in $\mathcal{G}$ is $\zeta(\Delta)$-colorable, where $\Delta := \Delta(G)$ denotes the maximum degree of G.*

For instance, the family of all graphs is an ζ-colorable family for the function $\zeta(\Delta) = \Delta+1$, and triangle-free graphs are ζ-colorable for $\zeta(\Delta) = O(\frac{\Delta}{\ln \Delta})$.

▶ **Theorem 31.** *Let $\mathcal{G}$ be a ζ-colorable family of graphs (see Definition 30) and $G(V, E)$ be an n-vertex graph with maximum degree Δ in $\mathcal{G}$. For the parameters*

$$\varepsilon > 0, \qquad 1 \leq k \leq \frac{\varepsilon^2 \cdot \Delta}{9 \ln n}, \qquad C := C(\varepsilon, k) = k \cdot \zeta\Big((1+\varepsilon) \cdot \frac{\Delta}{k}\Big),$$

suppose we partition V into k sets $V_1, \ldots, V_k$ uniformly at random; then with high probability G can be C-colored by coloring each $G[V_i]$ with a distinct palette of size C/k.

The proof of this theorem is by simply showing that the maximum degree of each graph $G[V_i]$ is sufficiently small, itself a simple application of Chernoff bound. As such, we postpone it to the full version of the paper.

B.1 Sublinear Algorithms from Theorem 31

As before, we only focus on streaming and query algorithms in this section. Table 2 contains a summary of our results in this part. In the following two algorithms, the parameters C and k are the same as in Theorem 31.

Algorithm 3 Streaming Algorithms from Theorem 31.

1. At the beginning, sample a random k-partitioning of the vertices into $V_1, \ldots, V_k$.
2. Throughout the stream, store any edge that belongs to one of the graphs $G[V_i]$.
3. At the end, use the stored subgraphs to find a C-coloring of G by coloring each $G[V_i]$ with a distinct palette of size C/k.

Using this algorithm and Theorem 31, we obtain the following corollary (proven formally in the full version).

Table 2 A sample of our sublinear algorithms obtained as corollaries of Theorem 31. All the streaming algorithms here are *single-pass* and all the sublinear-time algorithms are *non-adaptive*.

# of Colors	Graph Family	Streaming	Sublinear-Time
$O(\frac{\Delta}{\gamma \cdot \ln \Delta})$	Triangle-Free	$O(n\Delta^{2\gamma})$ space	$\widetilde{O}(n^{3/2+2\gamma})$ time
$O(\frac{\Delta \ln\ln \Delta}{\gamma \cdot \ln \Delta})$	K_r-Free	$O(n\Delta^{2\gamma})$ space	$\widetilde{O}(n^{3/2+\Theta(\gamma)})$ time
$O(\frac{\Delta}{\gamma \ln \Delta} \cdot \ln r)$	Locally r-Colorable	$O(n\Delta^{2\gamma})$ space	$\widetilde{O}(n^{3/2+2\gamma})$ queries
$O(\frac{\Delta}{\gamma \ln\ln n} \cdot \ln r)$	Locally r-Colorable	$O(n\Delta^{2\gamma})$ space	poly(n) time
$O(\frac{\Delta}{\ln(1/\delta)})$	δ-Sparse-Neighborhood	$O(n/\delta)$ space	$\widetilde{O}(n^{3/2} \cdot \text{poly}(1/\delta))$ time

▶ **Corollary 32.** *Let $\mathcal{G}$ be a ζ-colorable family of graphs (Definition 30). There exists a randomized streaming algorithm that makes a single pass over any graph G from $\mathcal{G}$ with maximum degree Δ, and for any setting of parameters:*

$$\varepsilon > 0, \qquad 1 \leq k \leq \frac{\varepsilon^2 \cdot \Delta}{9 \ln n}, \qquad C := C(\varepsilon, k) = k \cdot \zeta\Big((1+\varepsilon) \cdot \frac{\Delta}{k}\Big),$$

with high probability computes a proper C-coloring of G using $O(n \cdot \frac{\Delta}{k})$ space.

Algorithm 4 Query Algorithms from Theorem 31.

1. Sample a random k-partitioning of the vertices into $V_1, \ldots, V_k$.
2. Obtain the subgraphs $G[V_1], \ldots, G[V_k]$ using the following procedure:
   - If $\Delta > n/k$, then non-adaptively query all pairs of vertices u, v where both u, v belong to the same V_i (using pair queries);
   - Otherwise, non-adaptively query all neighbors of all vertices u (using neighbor queries).
3. Find a C-coloring of G by coloring each $G[V_i]$ with a distinct palette of size C/k (with no further access to G).

Again, using this algorithm and Theorem 31, we obtain the following corollary (proven formally in the full version).

▶ **Corollary 33.** *Let $\mathcal{G}$ be a ζ-colorable family of graphs (Definition 30). There exists a randomized non-adaptive algorithm that given query access to any graph G from $\mathcal{G}$ with maximum degree Δ, for any setting of parameters:*

$$\varepsilon > 0, \qquad 1 \leq k \leq \frac{\varepsilon^2 \cdot \Delta}{9 \ln n}, \qquad C := C(\varepsilon, k) = k \cdot \zeta\Big((1+\varepsilon) \cdot \frac{\Delta}{k}\Big),$$

with high probability computes a proper C-coloring of G using $\min\big\{O(n\Delta) + O(n^2/k)\big\}$ queries.

We conclude this section with some important remarks about Corollaries 32 and 33.

▶ Remark 34 (Runtime of our algorithms). We did not state the runtime of our algorithms in this section and focused primarily on space and query complexity of algorithms, respectively.

This is because in both cases, the runtime of the algorithm crucially depends on the runtime of the coloring algorithm for finding a ζ-coloring of each subgraph $G[V_i]$ which is specific to the family $\mathcal{G}$ (and ζ) and thus not known a-priori.

Nevertheless, for **almost all** our applications to specific families of graphs (with one exception), **the runtime of the algorithms is also sublinear in the input size**.

B.2 Particular Implications of Theorem 31

We now list the applications of Theorem 31 and Corollaries 32 and 33 to different families of "locally sparse" graphs that are colorable with much fewer than $(\Delta+1)$ colors.

Triangle-Free Graphs

As stated earlier, triangle-free graphs admit an $O(\frac{\Delta}{\ln \Delta})$ coloring. This was first proved by Johansson [21] by showing an upper bound of $9\frac{\Delta}{\ln \Delta}$ on the chromatic number of these graphs[5]. The leading constant was then improved to 4 by Pettie and Su [29] and very recently to $1+o(1)$ by Molloy [24] matching the result of Kim for graphs of girth 5 [23]. Moreover, Molloy's result implies an $\widetilde{O}(n\Delta^2)$ time algorithm for finding such a coloring.

Note that triangle-free graphs form a hereditary family of graphs and aforementioned results imply that they are $\zeta_{\textsf{tri-free}}$-colorable for $\zeta_{\textsf{tri-free}}(\Delta) = O(\frac{\Delta}{\ln \Delta})$. As such, Corollaries 32 and 33 imply the following algorithms for any $\gamma \in (0, 1/2)$ as small as $\Theta(\frac{\ln\ln\Delta}{\ln\Delta})$:

- **Streaming Model:** A randomized single-pass $\widetilde{O}(n^{1+\gamma})$ space algorithm for $O(\frac{\Delta}{\gamma \ln \Delta})$ coloring of triangle-free graphs. The post-processing time of this algorithm is $\widetilde{O}(n \cdot \Delta^{\gamma})$.
- **Query Model:** A randomized non-adaptive $\widetilde{O}(n^{3/2+\gamma})$-query algorithm for $O(\frac{\Delta}{\gamma \ln \Delta})$ coloring of triangle-free graphs. The runtime of this algorithm is also $\widetilde{O}(n^{3/2+2\gamma})$.

Both results above are proved by picking $\varepsilon = \Theta(1)$ and $k = \Theta(\Delta^{1-\gamma})$, thus obtaining a C-coloring:

$$C = C(\varepsilon, k) = k \cdot \zeta_{\textsf{tri-free}}\Big(\Theta(\Delta/k)\Big) = O(k) \cdot \frac{\Delta/k}{\ln(\Delta/k)} = O(\frac{\Delta}{\ln \Delta^{\gamma}}) = O(\frac{\Delta}{\gamma \ln \Delta}).$$

K_r-Free Graphs

For any fixed integer $r \geq 1$, we refer to any graph that does not contain a copy of the K_r, namely, the clique on r vertices, as a K_r-free graph. Johansson proved that any K_r-free graph admits an $O(\frac{\Delta \ln\ln\Delta}{\ln\Delta})$ coloring [22] and gave an $O(n \cdot \text{poly}(\Delta))$ time algorithm for finding it[6]. This result was very recently simplified (and extended to r beyond a fixed constant) by Molloy [24] (however the latter result does not imply an efficient algorithm).

Similar to the case of triangle-free graphs, combining these results with Corollaries 32 and 33 imply the following algorithms for any $\gamma \in (0, 1/2)$ as small as $\Theta(\frac{\ln\ln\Delta}{\ln\Delta})$:

- **Streaming Model:** A randomized single-pass $\widetilde{O}(n^{1+\gamma})$ space algorithm for $O(\frac{\Delta \ln\ln\Delta}{\gamma \ln\Delta})$ coloring of K_r-free graphs. The post-processing time of this algorithm is $O(n^{1+\Theta(\gamma)})$.
- **Query Model:** A randomized non-adaptive $\widetilde{O}(n^{3/2+\gamma})$-query algorithm for $O(\frac{\Delta \ln\ln\Delta}{\gamma \ln\Delta})$ coloring of K_r-free graphs. The runtime of this algorithm is also $O(n^{3/2+\Theta(\gamma)})$.

[5] This result of Johansson was never published – see [26, Chapter 13] for a lucid presentation of the original proof.

[6] This result of Johansson was also never published – see [6] for a streamlined version of this proof.

Graphs with r-Colorable Neighborhoods

For any fixed integer $r \geq 1$, we say that a graph G is locally r-colorable iff neighborhood of every vertex in G is r-colorable. Johansson also proved that r-colorable graphs admits an $O(\frac{\Delta}{\ln \Delta} \cdot \ln r)$ coloring [22]; see [6] for a proof and also an algorithm that finds such a coloring in $\text{poly}(n \cdot 2^{\Delta})$ time (which uses, as a subroutine, a result of [9]).

It is easy to see that locally r-colorable graphs also form a hereditary family. Consequently, as before, Corollaries 32 and 33 imply the following for any $\gamma \in (0, 1/2)$ as small as $\Theta(\frac{\ln \ln \Delta}{\ln \Delta})$:

- **Streaming Model:** A randomized single-pass $\widetilde{O}(n^{1+\gamma})$ space algorithm for $O(\frac{\Delta}{\gamma \ln \Delta} \cdot \ln r)$ coloring of locally r-colorable graphs. The post-processing time of the algorithm is $\text{poly}(n \cdot 2^{\Delta^{\gamma}})$.
- **Query Model:** A randomized non-adaptive $\widetilde{O}(n^{3/2+\gamma})$-query algorithm for $O(\frac{\Delta}{\gamma \ln \Delta} \cdot \ln r)$ coloring of locally r-colorable graphs. The runtime of this algorithm is also $\text{poly}(n \cdot 2^{\Delta^{\gamma}})$.

Graphs with δ-Sparse Neighborhoods

For any $\delta \in (0, 1)$, we say a graph $G(V, E)$ has a δ-sparse neighborhood iff the total number of edges in the neighborhood of any vertex v (i.e., edges between neighbors of v) is at most $\delta \cdot \Delta^2$ (not to be confused with Definition 14 for ε-sparse vertices, albeit the two definitions are equivalent for Δ-*regular* graphs by setting $\delta = (1 - \varepsilon^2)$). Alon, Krivelevich and Sudakov [2] proved that any graph G with maximum degree Δ and δ-sparse neighborhood admits an $O(\frac{\Delta}{\log (1/\delta)})$ coloring and that this is tight for all admissible values of δ and Δ.

We note that unlike all other families of graphs considered in this section, the family of sparse-neighborhood graphs is *not* a hereditary family. As such, we cannot readily apply Theorem 31 (and hence Corollaries 32 and 33). However, we can modify the proof of Theorem 31 slightly to apply to this case as well (see the full version).

▶ **Lemma 35.** *For any $\delta \in (0, 1)$, let $G(V, E)$ be an n-vertex graph with maximum degree Δ and δ-sparse neighborhoods. For the parameters*

$$1 \leq k \leq \frac{\delta \cdot \Delta}{9 \cdot \ln n}, \qquad C := \Theta(\frac{\Delta}{\ln (1/\delta)}),$$

suppose we partition V into k sets $V_1, \ldots, V_k$ uniformly at random; then with high probability G can be C-colored by coloring each $G[V_i]$ with a distinct palette of size C/k.

Similar to Corollaries 32 and 33, this in turn implies the following algorithms:

- **Streaming Model:** A randomized single-pass $\widetilde{O}(n/\delta)$ space algorithm for $O(\frac{\Delta}{\ln (1/\delta)})$ coloring of graphs with δ-sparse neighborhoods. The post-processing time is $\widetilde{O}(n \cdot \text{poly}(1/\delta))$.
- **Query Model:** A randomized non-adaptive $\widetilde{O}(n^{3/2}/\delta)$-query algorithm for $O(\frac{\Delta}{\ln (1/\delta)})$ coloring of graphs with δ-sparse neighborhoods. The runtime of the algorithm is $\widetilde{O}(n^{3/2} \cdot \text{poly}(1/\delta))$

On Hitting-Set Generators for Polynomials That Vanish Rarely

Dean Doron
Department of Computer Science, Stanford University, CA, USA
ddoron@stanford.edu

Amnon Ta-Shma
The Blavatnik School of Computer Science, Tel-Aviv University, Israel
amnon@tau.ac.il

Roei Tell
Department of Computer Science and Applied Mathematics, Weizmann Institute of Science, Rehovot, Israel
roei.tell@weizmann.ac.il

Abstract

The problem of constructing hitting-set generators for polynomials of low degree is fundamental in complexity theory and has numerous well-known applications. We study the following question, which is a relaxation of this problem: Is it easier to construct a hitting-set generator for polynomials $p\colon \mathbb{F}^n \to \mathbb{F}$ of degree d if we are guaranteed that the polynomial vanishes on at most an $\varepsilon > 0$ fraction of its inputs? We will specifically be interested in tiny values of $\varepsilon \ll d/|\mathbb{F}|$. This question was first considered by Goldreich and Wigderson (STOC 2014), who studied a specific setting geared for a particular application, and another specific setting was later studied by the third author (CCC 2017).

In this work our main interest is a *systematic study of the relaxed problem*, in its general form, and we prove results that significantly improve and extend the two previously-known results. Our contributions are of two types:

- Over fields of size $2 \leq |\mathbb{F}| \leq \text{poly}(n)$, we show that the seed length of any hitting-set generator for polynomials of degree $d \leq n^{.49}$ that vanish on at most $\varepsilon = |\mathbb{F}|^{-t}$ of their inputs is at least $\Omega\left((d/t) \cdot \log(n)\right)$.
- Over $\mathbb{F}_2$, we show that there exists a (non-explicit) hitting-set generator for polynomials of degree $d \leq n^{.99}$ that vanish on at most $\varepsilon = |\mathbb{F}|^{-t}$ of their inputs with seed length $O\left((d-t) \cdot \log(n)\right)$. We also show a polynomial-time computable hitting-set generator with seed length $O\left((d-t) \cdot \left(2^{d-t} + \log(n)\right)\right)$.

In addition, we prove that the problem we study is closely related to the following question: "Does there exist a small set $S \subseteq \mathbb{F}^n$ whose degree-d closure is very large?", where the degree-d closure of S is the variety induced by the set of degree-d polynomials that vanish on S.

2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

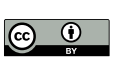
Keywords and phrases Hitting-set generators, Polynomials over finite fields, Quantified derandomization

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.7

Category RANDOM

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/119/`.

Funding *Dean Doron*: Supported by a Motwani Postdoctoral Fellowship and by NSF award CCF-1763311.
Amnon Ta-Shma: Supported by ISF grant 18/952 and by Len Blavatnik and the Blavatnik Family foundation.



© Dean Doron, Amnon Ta-Shma, and Roei Tell;
licensed under Creative Commons License CC-BY



Roei Tell: Supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 819702).

Acknowledgements We are grateful to an exceptionally helpful anonymous reviewer, who pointed us to the work of Nie and Wang [33] and to follow-up works, suggested an alternative proof strategy for the lower bound in the special case of prime fields (the alternative proof for this special case appears in the full version), and helped improve our initial results.

1 Introduction

Let $\mathcal{P}_{n,q,d}$ denote the set of all polynomials $\mathbb{F}^n \to \mathbb{F}$ of total degree d over the field of size $q = |\mathbb{F}|$. We think of n as sufficiently large, and of the degree d and the field size q as functions of n. For simplicity, throughout the paper we assume that $d < n$.[1]

A fundamental problem in complexity theory is that of constructing *hitting-set generators for low-degree polynomials*. Recall that a Hitting-Set Generator (HSG) for $\mathcal{P}_{n,q,d}$ is a function $H\colon \{0,1\}^\ell \to \mathbb{F}^n$ such that for every non-zero polynomial $p \in \mathcal{P}_{n,q,d}$ there exists $s \in \{0,1\}^\ell$ satisfying $p(H(s)) \neq 0$ (see Definition 11); in other words, every non-zero polynomial $p \in \mathcal{P}_{n,q,d}$ does not vanish on at least one element in the *hitting-set* $S = \left\{H(s)\colon s \in \{0,1\}^\ell\right\}$. The two main measures of efficiency for HSGs are the seed length ℓ (equivalently, the size of the hitting-set S as a multiset) and the computational complexity of H as a function (i.e., the computational complexity of generating an element of the hitting-set S given its index s).

A standard linear-algebraic argument yields a lower bound of $\Omega\left(d \cdot \log\left(n/d\right)\right)$ on the seed length of any HSG for $\mathcal{P}_{n,q,d}$, and a standard probabilistic argument shows that there *exists* a HSG for $\mathcal{P}_{n,q,d}$ with matching seed length $O\left(d \cdot \log(n/d) + \log\log(q)\right)$ (see Fact 14 and Fact 15). Naturally, the probabilistic upper-bound does not guarantee that the function H is *efficiently-computable*. Thus, the main open problem concerning HSGs for $\mathcal{P}_{n,q,d}$ is to construct efficiently-computable HSGs with seed length that matches the known lower bound. This well-known problem (as well as a variant that refers to *pseudorandom generators* as in Definition 13) has attracted a significant amount of attention over the years; see, e.g., [32, 29, 26, 25, 9, 10, 8, 27, 43, 28, 12, 35], and the related survey by Viola [42].

Several years ago, Goldreich and Wigderson [18, Section 5] considered a *relaxed version* of the foregoing problem. In general terms, what they asked is the following:

> Does the HSG problem become easier if we are guaranteed that the polynomial *vanishes rarely* (i.e., has very few roots)?

Note that, intuitively, we expect that the relaxed problem will indeed be easier: This is both since there are less polynomials that vanish rarely (than arbitrary polynomials), and since for any such polynomial p, almost all inputs will "hit" p.

In their original paper, Goldreich and Wigderson considered a specific instance of this problem, geared for a particular application (see Section 1.2 for details). In this paper our goal is to *study the relaxed problem in and of itself, in a systematic and general way*. Our motivation for doing so is three-fold. First, this is a special (and potentially-easy) case of the classical HSG problem, and thus constitutes a potential path to make progress on the classical problem. Secondly, the relaxed question is of independent interest as part of the broad study of *quantified derandomization*, which was initiated in the original work of Goldreich and Wigderson [18] (see also, e.g., [40, 11, 14]). And thirdly, as polynomial-based

[1] Most of our results also carry on to the setting of $d > n$, albeit with less "clean" parametrizations.

constructions are ubiquitous in complexity theory, any progress in our understanding of structured classes of polynomials or in related HSG constructions may be valuable for other explicit constructions.

To be more formal, denote by $\mathcal{P}_{n,q,d,\varepsilon}$ the set of polynomials $p \in \mathcal{P}_{n,q,d}$ such that $\Pr_{x\in\mathbb{F}^n}[p(x)=0] \leq \varepsilon$; that is, $\mathcal{P}_{n,q,d,\varepsilon}$ is the set of degree-d polynomials that *vanish rarely*, where the notion of "rarely" is parametrized by the parameter ε. The two main questions we consider in this context are:

- **The combinatorial question:** What is the minimal size of a hitting-set for $\mathcal{P}_{n,q,d,\varepsilon}$? Equivalently, we ask what is the minimal seed length of any HSG for $\mathcal{P}_{n,q,d,\varepsilon}$. This question is combinatorial since it refers to the *existence* of a HSG, regardless of its computational complexity.
- **The computational question:** For which values of $\varepsilon > 0$ can we construct a HSG for $\mathcal{P}_{n,q,d,\varepsilon}$ with small seed length that will be *efficiently-computable*? In other words, can we simultaneously optimize not only the seed length but also the *computational complexity* of HSGs for $\mathcal{P}_{n,q,d,\varepsilon}$?

1.1 Context and Previous Work

Let us first delineate some trivial values for ε. To do so, first recall that we expect a random polynomial to vanish on q^{-1} of its inputs. Now, by the Schwartz-Zippel lemma, any non-zero $p \in \mathcal{P}_{n,q,d}$ has at most an $\varepsilon = d/q$ fraction of roots; this bound is quite good when q is large compared to d, and in general, for arbitrary d and q, any non-zero polynomial vanishes on at most $1-\delta$ of its inputs, where $\delta \geq q^{-d/(q-1)}$ denotes the relative distance of the Reed-Muller code of degree d over $\mathbb{F}_q$. Therefore, the value $\varepsilon = 1-\delta$ represents the general case (i.e., the case of hitting *any* non-zero polynomial). Remarkably, we also have a minimal non-zero value that ε can have: By a theorem of Warning [45], every polynomial in $\mathbb{F}_q^n \to \mathbb{F}_q$ of degree d that vanishes *somewhere* vanishes on at least a q^{-d} fraction of its inputs. Therefore, hitting polynomials that vanish on $\varepsilon < q^{-d}$ fraction of their inputs is trivial, since such polynomials have no zeroes. It will be useful to denote $\varepsilon = q^{-t}$ from now on.

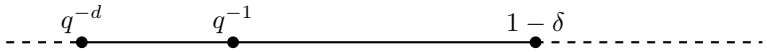


Figure 1 The two extremal values of ε (i.e., $\varepsilon = q^{-d}$ and $\varepsilon = 1-\delta$) and the expected $\varepsilon = q^{-1}$ for a random polynomial. (The parameter δ denotes the relative distance of the corresponding q-ary Reed-Muller code $RM(n,d)$.)

Referring to the combinatorial question, the standard probabilistic argument mentioned before shows there exists a HSG for $\mathcal{P}_{n,q,d,\varepsilon}$ with seed length $O(\log\log(|\mathcal{P}_{n,q,d,\varepsilon}|))$. Thus, the combinatorial question is intimately connected to the long-standing open problem of determining the *weight distribution of the Reed-Muller code*, i.e., *counting* the number of polynomials in $\mathcal{P}_{n,q,d}$ that vanish on precisely $\varepsilon > 0$ of their inputs, for every $\varepsilon > 0$. The latter problem has been studied since the late 60's (see, e.g., [4, 22]), but is currently settled only for $d = 2$ (see [37, 31]). Only recently have general results been obtained for $d > 2$, and the bounds in these results are asymptotic (rather than precise bounds) and hold only over $\mathbb{F}_2$ (see [24, 1]). More generally, this problem is a special case of the well-known problem of studying weight distributions of (classes of) linear codes, which is typically tackled using weight enumerator polynomials (for relevant background see, e.g., [30, Chapter 5]). Note, however, that the weight distribution problem is more general, since it refers to all non-trivial values of $\varepsilon > 0$, whereas in our setting we focus only on tiny values of ε.

Another related line of works focuses on structural properties of *biased polynomials*. Fixing a polynomial $p\colon \mathbb{F}^n \to \mathbb{F}$ and looking at the distribution over $\mathbb{F}$ that is obtained by evaluating p at a random point, we can ask whether this distribution is close to uniform, or whether it is far from uniform, in which case we call the polynomial biased. A sequence of works showed that biased polynomials are very "structured", in the sense that they can be determined by a relatively-small number of polynomials of lower degree (see [19, 23, 21, 5, 7, 6]). Our setting is much more specific than the setting in these works, since their assumption is only that the polynomial is *biased*, whereas our assumption is that the polynomial is biased in a very specific manner (i.e., one output-value has tiny weight $\varepsilon > 0$). Thus, the results in these works typically do not seem sufficiently strong to be useful in our more specific setting.[2]

Goldreich and Wigderson [18, Section 5], who were motivated by a specific application in circuit complexity (derandomization of $\mathcal{AC}^0[\oplus]$), constructed a polynomial-time computable HSG for the setting of $q = 2$ and $\varepsilon = 2^{-(d-O(1))} = O(2^{-d})$ (for details see Section 1.2). Thus, they gave an upper-bound for the *computational question*, which holds only for $\mathbb{F}_2$ polynomials with extremely few roots. In a subsequent work by the third author [40], two combinatorial lower bounds were proved for the setting of $q = \text{poly}(n)$ and $\varepsilon = q^{-O(1)}$ (again, for details see Section 1.2). Thus, the subsequent work showed lower bounds for the *combinatorial question*, which hold only for polynomials over $\mathbb{F}_{\text{poly}(n)}$ with a relatively-large number of roots (i.e., only mildly less roots than the expected value of $\varepsilon = q^{-1}$). In both previous works, ad-hoc arguments were used to obtain the corresponding results.

1.2 Our Main Results

Our first main result is a general lower bound for the combinatorial problem. For context, in [40] it was shown that when $q = \text{poly}(n)$, any HSG for $\mathcal{P}_{n,d,q,q^{-O(1)}}$ requires a seed of length $\Omega(d^{\Omega(1)} \cdot \log(n/d^{\Omega(1)}))$; and any HSG with constant density[3] for $\mathcal{P}_{n,d,q,q^{-1}}$ requires a seed of length $\Omega(d \cdot \log(n/d))$. Thus, both previous lower bounds referred to the setting of $q = \text{poly}(n)$ and of $\varepsilon = q^{-O(1)}$ (i.e., $t = O(1)$).

The following result shows a lower bound that is both significantly stronger, and – more importantly – applies to a far broader parameter setting. In particular, the following result applies to a general $q \leq \text{poly}(n)$ and to values of $\varepsilon = q^{-t}$ almost up to the extreme value of $\varepsilon = q^{-d}$, and gives a lower bound of $\Omega((d/t) \cdot \log(n))$:

▶ **Theorem 1** (lower bound over general fields). *For every constant $c > 1$ there exists a constant $\gamma > 0$ such that the following holds. For every $n, q, d, t \in \mathbb{N}$ such that $2 \leq q \leq n^c$ is a prime power, $d \leq n^{.49}$, and $t \leq \gamma \cdot d$, any HSG for $\mathcal{P}_{n,q,d,q^{-t}}$ requires a seed of length $\Omega\left((d/t) \cdot \log(n)\right)$.*

Let us parse the meaning of the lower bound in Theorem 1. For comparison, recall that there exists a HSG for all polynomials of degree $d \leq n^{.49}$ with seed length $O(d \cdot \log(n))$. Theorem 1 tells us that the relaxation of only requiring to "hit" polynomials that vanish with probability q^{-t} can "buy" a factor of at most $1/t$ in the seed length. In particular, there does not exist a significantly smaller hitting-set for polynomials that vanish with probability $q^{-O(1)}$. Perhaps surprisingly, this is also true for polynomials that vanish with probability $q^{-d^{o(1)}}$ (since the lower bound remains almost linear in $d \cdot \log(n)$). Only for polynomials that vanish with probability $q^{-d^{\Omega(1)}}$ does our lower bound imply that a significantly smaller

[2] One exception is the field $\mathbb{F}_2$, in which the notions of bias and of "vanish rarely" converge. Indeed, the proofs of our results for $\mathbb{F}_2$ use insights developed in this sequence of works.

[3] A hitting-set S for a class $\mathcal{P}$ has density $\varepsilon > 0$ if for every $p \in \mathcal{P}$ it holds that $\Pr_{s \in S}[p(s) \neq 0] \geq \varepsilon$.

hitting-set *might* exist; and at an "extreme" value of $q^{-\Omega(d)}$, our lower bound does not rule out a polynomial-sized hitting-set. For technical statements that include various extensions and improvements of Theorem 1 (and in particular also hold for polynomials of higher degree $n^{.49} < d \leq \gamma \cdot n$), see Section 5.[4]

Now, still referring to the combinatorial question, we observe that a result of Kaufmann, Lovett, and Porat [24], which upper-bounds the *number* of biased $\mathbb{F}_2$ polynomials (i.e., analyzes the weight distribution of the Reed-Muller code over $\mathbb{F}_2$), yields a corresponding existential upper-bound. Specifically:

▶ **Theorem 2** (upper-bound over $\mathbb{F}_2$, following [24]). *Let $n, d, t \in \mathbb{N}$ where $d > t$. Then, there exists a (non-explicit) hitting-set for $\mathcal{P}_{n,2,d,2^{-t}}$ with seed length $O\left((d-t) \cdot \log(\frac{n}{d-t})\right)$.*

Note that while the lower bound in Theorem 1 holds for any finite field, the upper bound in Theorem 2 holds only over $\mathbb{F}_2$. Nevertheless, comparing Theorem 1 and Theorem 2 (for $\mathbb{F} = \mathbb{F}_2$ and $d \leq n^{.49}$) reveals that there is still a *significant gap* between the upper-bound and the lower-bound: The lower bound is of the form $(d/t) \cdot \log(n)$, whereas the existential upper bound is of the form $(d-t) \cdot \log(n)$. For example, the lower bound indicates that there *might* exist a significantly smaller hitting-set for the relaxed problem when $t = d^{\Omega(1)}$, whereas the existential upper bound is significantly better than the one for the original problem only for $t = d - d^{\Omega(1)}$.

Our last main result is computational and shows an *explicit* construction of a HSG. As mentioned above, Goldreich and Wigderson [18] constructed a polynomial-time computable HSG with seed length $O(\log(n))$ that "hits" polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ of degree d that vanish on $O(2^{-d})$ of their inputs (for any $d \in \mathbb{N}$). We prove a significantly more general result, by constructing an explicit HSG for $\mathcal{P}_{n,2,d,2^{-t}}$ for any $t < d - O(1)$:

▶ **Theorem 3** (explicit upper-bound over $\mathbb{F}_2$). *Let $n \in \mathbb{N}$ be sufficiently large, and let $d > t+4$ be integers. Then, there exists a polynomial-time computable HSG for $\mathcal{P}_{n,2,d,2^{-t}}$ with seed length $O\left((d-t) \cdot \left(2^{d-t} + \log(\frac{n}{d-t})\right)\right)$.*

Note that the original result from [18] is the special case of Theorem 3 when $t = d - O(1)$. Also note that the seed length of the explicit HSG from Theorem 3 depends exponentially on $d-t$, whereas the seed length of the non-explicit HSG from Theorem 2 depends linearly on $d-t$. We also comment that the result is actually slightly stronger, and asserts that for any $r \in \mathbb{N}$ there exists a polynomial-time computable HSG for $\bigcup_d \mathcal{P}_{n,2,d,q^{d-r}}$ with seed length $O(r \cdot (2^r + \log(n/r)))$; that is, for every r there is a *single* HSG that works for *all degrees* d with $t = d - r$.

Below, in Table 1, we present an informal summary of the main results mentioned above, and compare them to previously-known results.

1.3 The Connection to Small Sets With Large Degree-d Closures

In addition to our lower-bounds and upper-bounds for the problem of HSGs for polynomials that vanish rarely, we also tie this problem to the study of a clean and elegant algebraic question; namely, to the study of the degree-d closure of a set $S \subseteq \mathbb{F}^n$, which was recently initiated by Nie and Wang [33].

[4] In these technical results, the $\log(n)$ term in the lower bound in Theorem 1 is replaced by a more complicated term that depends on d and on t, for example $\log(n^{.99} \cdot (t/d))$.

Table 1 An informal summary of our results and comparison to previous results.

	Seed length		Field Size	ε
Lower bounds				
[40]	$\Omega(d^{\Omega(1)} \cdot \log(n/d^{\Omega(1)}))$		$q = \text{poly}(n)$	$q^{-O(1)}$
Theorem 1	$\Omega((d/t) \cdot \log n)$	$(d \leq n^{.49})$	$2 \leq q \leq \text{poly}(n)$	q^{-t}
Theorem 23	$\Omega((d/t) \cdot \log(n^{.99} \cdot t/d))$	$(d/t \lessapprox q \cdot n^{.01})$	$2 \leq q \leq \text{poly}(n)$	q^{-t}
Upper bounds				
[18]	$O(\log n)$	(explicit)	$q = 2$	$2^{-d+O(1)}$
Theorem 2	$O((d-t)\log(\frac{n}{d-t})$	(non-explicit)	$q = 2$	2^{-t}
Theorem 3	$O((d-t) \cdot (2^{d-t} + \log(\frac{n}{d-t}))$	(explicit)	$q = 2$	2^{-t}

Using terminology from algebraic geometry, the degree-d closure of a set $S \subseteq \mathbb{F}^n$ is a finite-degree analogue of the Zariski closure of S, and is defined as the variety induced by the set of degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$ that vanish on S. In more detail, let us first define the degree-d ideal of S to be $\mathcal{I}^{(d)}(S) = \{p \in \mathcal{P}_d : \forall s \in S, p(s) = 0\}$, where $\mathcal{P}_d$ is the set of degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$.[5] Then, the degree-$d$ closure of S is defined by:

$$\texttt{Cl}^{(\texttt{d})}(S) = \{x \in \mathbb{F}^n : \forall p \in \mathcal{I}^{(d)}(S), p(x) = 0\}.$$

As an example, observe that the degree-d closure of any $d+1$ points on a fixed line in $\mathbb{F}^n$ contains the entire line. As another example, recall that the closure of any Kakeya set in $\mathbb{F}_q^n$ with respect to homogeneous degree-$(q-1)$ polynomials is the entire domain $\mathbb{F}_q^n$ (this was proved by Dvir [16, Section 3] towards showing that any Kakeya set is necessarily of size at least $\binom{q+n-1}{n}$).

Following the latter example, it is natural to ask whether there exists a *very small* set $S \subseteq \mathbb{F}^n$ whose degree-d closure is *very large*. An initial observation towards answering this question is that a set $S \subseteq \mathbb{F}^n$ has *maximal* degree-d closure (i.e., $\texttt{Cl}^{(\texttt{d})}(S) = \mathbb{F}^n$) if and only if S is a hitting-set for degree-d polynomials. (This is since in both cases, the only degree-d polynomial that vanishes on S is the zero polynomial.)

▶ **Observation 4** (maximal closure $\iff$ hitting-set)**.** *A set $S \subseteq \mathbb{F}^n$ is a hitting-set for (all) degree-d polynomials* if and only if $\left|\texttt{Cl}^{(\texttt{d})}(S)\right| = q^n$.

Loosely speaking, the main result of Nie and Wang [33] extends Observation 4 by showing that that for any $S \subseteq \mathbb{F}^n$ it holds that $\left|\texttt{Cl}^{(\texttt{d})}(S)\right| \leq \frac{|S|}{\binom{n+d}{d}} \cdot |\mathbb{F}|^n$. The meaning of this result is that, while there exist sets of size $|S| = \binom{n+d}{d}$ whose degree-d closure is $\mathbb{F}^n$, the degree-d closure of smaller sets decreases by a factor of at least $\frac{|S|}{\binom{n+d}{d}}$.[6]

[5] Note that $\mathcal{I}^{(d)}(S)$ is not an actual ideal in the ring of n-variate polynomials over $\mathbb{F}$, since multiplying $p \in \mathcal{I}^{(d)}(S)$ by another polynomial does not necessarily preserve the degree of p.

[6] Another result along these lines was recently proved by Beelen and Datta [3], who showed a tight upper-bound on the size of the variety induced by *any* subspace of degree-d polynomials (rather than only for varieties induced by a subspace of the form $\mathcal{I}^{(d)}(S)$ for some $S \subseteq \mathbb{F}^n$).

We take another approach to extending Observation 4, by establishing a connection between the study of small sets with large closures and the study of HSGs for polynomials that vanish rarely. Specifically, we show two-way implications between the statement that S is a hitting-set generator for polynomials *that vanish rarely*, and the statement that S has *large closure*. In more detail, we relate hitting-sets for polynomials that vanish with probability q^{-t} to sets with closure of size q^{n-t}:

▶ **Theorem 5** (small sets with large closures versus hitting-sets for polynomials that vanish rarely). *Let $\mathbb{F}$ be a field of size q, let $n \in \mathbb{N}$ and $t < d < n$, and let $S \subseteq \mathbb{F}^n$. Then,*

1. *If $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| > q^{n-t}$, then S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$.*
2. *If S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$, then $\left|\mathtt{Cl}^{(\mathtt{d}/2(\mathtt{t}+1))}(S)\right| > \frac{1}{2} \cdot q^{n-t}$.*

Notice that Theorem 5 does not show a complete equivalence between the two notions, since in the second item the closure refers to degree $d/2t$ rather than to degree d. Thus, intuitively, Theorem 5 asserts that constructing a small set with a large degree-d closure is at least as hard as constructing a hitting-set for polynomials that vanish rarely; and while it also gives a converse reduction (in the second item), it is nevertheless possible that constructing a hitting-set for polynomials that vanish rarely is an easier problem. We also remark that the first item in Theorem 5 is almost immediate, whereas the second item requires more work (see Appendix C for details).

Lastly, we comment that one can obtain an upper-bound on the size of $\mathtt{Cl}^{(\mathtt{d})}(S)$ for small sets $S \subseteq \mathbb{F}^n$ by combining the first item in Theorem 5 with our lower bound from Theorem 1. (This is since the former asserts that sets with closure of size q^{n-t} are hitting-sets for $\mathcal{P}_{n,q,d,q^{-t}}$, whereas the latter asserts that any such hitting-set must be large.) However, the bounds obtained in this way are not stronger than the known bounds proved in [33]. For more details see Appendix C.

2 Overview of Our Techniques

2.1 Combinatorial Lower Bounds From Low-Degree Dispersers

The proofs of our lower bounds on HSGs for polynomials that vanish rarely rely on a *complexity-theoretic* approach, rather than on a direct algebraic analysis. Specifically, we reduce the problem of constructing HSGs for *arbitrary* polynomials to the problem of constructing HSGs for polynomials that *vanish rarely*; since we already know lower bounds for the former, we obtain lower bounds for the latter.

Specifically, given an arbitrary non-zero polynomial $p_0 \colon \mathbb{F}^m \to \mathbb{F}$, we will use a form of "error-reduction" for polynomials (akin to error-reduction for probabilistic algorithms; see below) to obtain another polynomial $p \colon \mathbb{F}^n \to \mathbb{F}$ such that:

1. The polynomial p vanishes rarely.
2. Any non-zero input for p can be mapped into a small list of inputs for p_0 that contains a non-zero input for p_0.

To define p, fix a (k,δ)-disperser $\mathsf{Disp} \colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$, for appropriate parameters k and δ that we will determine in a moment.[7] Then, p is the result of the following procedure: Given $z \in \mathbb{F}^n$, compute the 2^ℓ inputs $\{\mathsf{Disp}(z,i)\}_{i \in \{0,1\}^\ell}$, evaluate p_0 at each of these inputs, and output the disjunction of these evaluations; that is:

$$p(z) = \bigvee_{i \in \{0,1\}^\ell} p_0\left(\mathsf{Disp}(z,i)\right).$$

[7] A (k,δ)-disperser $\mathsf{Disp} \colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ is a function such that for every $T \subseteq \mathbb{F}^m$ satisfying $|T|/|F|^m \geq \delta$, for all but at most 2^k of the inputs $z \in \mathbb{F}^n$ there exists $i \in \{0,1\}^\ell$ such that $\mathsf{Disp}(z,i) \in T$.

The disperser Disp has the property that for every set $T \subseteq \mathbb{F}^m$ of density at least δ it holds that $\Pr_{z\in\mathbb{F}^n}[\forall i\ \mathsf{Disp}(z,i) \notin T] \leq \varepsilon = 2^k/q^n$. We take T to be the set of elements in $\mathbb{F}^n$ on which p_0 does not vanish, and take δ to be the density of T (i.e., δ is the distance of the corresponding Reed-Muller code); we also let $k = (n-t)\cdot\log(q)$. Then, the polynomial p vanishes on at most an $\varepsilon = 2^k/q^n = q^{-t}$ fraction of its inputs. Also, any non-zero input $z \in \mathbb{F}^n$ for p can be mapped to a list of 2^ℓ inputs $\{x_i = \mathsf{Disp}(z,i)\}_{i\in\{0,1\}^\ell}$ for p_0 such that for some $i \in \{0,1\}^\ell$ it holds that $p_0(x_i) \neq 0$, as we wanted.

The reduction above shows that if there exists a HSG with seed length s for polynomials $\mathbb{F}^n \to \mathbb{F}$ of degree $d = \deg(p)$ that vanish with probability ε, then there exists a corresponding HSG with seed length $s + \ell$ for all non-zero polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree $d_0 = \deg(p_0)$. The known lower bound on the latter, which asserts that $s + \ell = \Omega(d_0 \cdot \log(m/d_0))$, yields a corresponding lower bound on the former.

While this is indeed our main idea, it unfortunately does not quite work as-is. The main challenge is that the reduction above incurs *significant overheads* that crucially deteriorate the lower bound. Most importantly, the *degree* of the polynomial increases (from $d_0 = \deg(p_0)$ to $d = \deg(p)$), and the number of variables also increases (from m to n); this affects us since we are interested in a lower bound as a function of n and d, whereas our lower bound is a function of m and d_0. Moreover, the lower bound deteriorates by an additive factor of ℓ, since each non-zero input $z \in \mathbb{F}^n$ for p yields 2^ℓ inputs for p_0, one of which is guaranteed to be non-zero. Thus, we want to modify the reduction above, in order to minimize the blowup in the degree and in the number of variables, and also minimize the seed length ℓ of the disperser.

A coding-theoretic perspective

One can view the procedure described above as amplifying the *weight* (i.e., the fraction of non-zero coordinates) of a codeword in the Reed-Muller code. At first glance, this task seems similar to the task of amplifying the *distance* of linear error-correcting codes; in particular, the disperser-based technique described above is technically reminiscent of the well-known distance amplification technique of Alon *et al.* [2].[8] However, the crucial difference is that we are interested in amplifying the weight to be much larger than $1 - 1/q$, and indeed our resulting subcode (of polynomials that vanish rarely) is a small and non-linear subcode of the Reed-Muller code. Moreover, as explained above, we will be particularly interested in the degree blow-up, which is a parameter specific to polynomial-based codes.

Warm-up: The setting of $d \ll q$

For simplicity, let us assume that $q = \mathrm{poly}(n)$ and that $d \leq n^{.99}$. In this case the fraction δ of non-zeroes of p_0 is very close to one and we only need Disp to be a $(k, .99)$-disperser for $k = (n-t)\cdot\log(q)$.

Note that to compute p at an input $z \in \mathbb{F}^n$, we wish to compute $\mathsf{Disp}_i(z) = \mathsf{Disp}(z,i)$ as a function of z for each *fixed* value i of the seed. Since we want p to have degree as low as possible, we are interested in objects that we call low-degree dispersers: Informally, a disperser $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ has low degree if for any $i \in \{0,1\}^\ell$ and $j \in [m]$, the polynomial $q_{i,j}(z) = \mathsf{Disp}(z,i)_j$ (i.e., $q_{i,j}(z)$ is the j^{th} output element of $\mathsf{Disp}(z,i)$ as a function of z)

[8] The main differences are that we will use a specific disperser that is different from theirs, to minimize the degree blow-up; and that we handle alphabet reduction differently (using an `OR` function instead of code concatenation), since our target weight is much larger than $1 - 1/q$.

has low degree (see Definition 16 and Definition 17). Note that in our argument we only need the *existence* of a low-degree disperser (i.e., we do not need the low-degree disperser to be efficiently computable); however, the dispersers that are obtained via naive probabilistic arguments do not have low degree.

Fortunately, in the current "warm-up" setting we can get a good (albeit non-optimal) lower bound even using the "naive disperser" that just performs uniform sampling: That is, the disperser that treats its input $z \in \mathbb{F}^n$ as n/m substrings of length m, and treats its seed as an index $i \in [n/m]$, and outputs the i^{th} substring of length m in z. Note that this disperser is *linear* (i.e., has degree one), since for a fixed seed, each output element is a projection of a corresponding input element.

We do encounter one other problem in implementing our idea in this setting, which is the degree blow-up that comes from the fact that p computes the OR function on the outputs of the disperser (recall that the OR function of 2^ℓ inputs has maximal degree $(q-1)\cdot 2^\ell$). To circumvent this problem, we replace the OR function with a multivalued OR function. Specifically, observe that in the reduction above it suffices that on any non-zero input $y \in \mathbb{F}^{2^\ell}$, the OR function will output *some* non-zero element (rather than map any non-zero y to $1 \in \mathbb{F}$). In contrast to the OR function, there exists a multivalued OR function of 2^ℓ elements with degree roughly 2^ℓ (see Proposition 10).

Working out the precise parameters, this approach transforms any p_0 of degree d_0 into a corresponding p of degree $d = d_0 \cdot 2^\ell = d_0 \cdot t \cdot \log(q)$, and for every $t \leq d/O(\log(q))$ implies a lower bound of $\Omega(d_0 \cdot \log(m/d_0)) - \ell = \Omega(d/t)$ on the seed length of HSGs for polynomials that vanish with probability q^{-t}. To improve this lower bound to match the bound stated in Theorem 1, we use a disperser that is better than the naive one, and utilize the techniques that are outlined below (see Section 5).

The more challenging setting of $d \gg q$

Observe that in the argument above we "paid" for the seed length ℓ of the disperser *twice*: One loss was a blow-up of 2^ℓ in the degree (since the multivalued OR function has degree 2^ℓ), and the other loss was that the lower bound on the seed length of the HSG decayed additively in ℓ (because our reduction maps any non-zero input for p to a list of 2^ℓ inputs for p_0). Also note that the first loss decreases the lower bound itself, whereas the second loss limits the values of t to which the lower bound applies (to ones for which $\ell \ll d_0 \cdot \log(m/d_0)$).

When $d \gg q$ these two losses may deteriorate our lower bound much more severely than in the "warm-up" setting. This is because when q was large we instantiated the disperser with the parameter $\delta = \Omega(1)$, and hence its seed length was relatively small, whereas in our current setting the value of $\delta = q^{-d_0/(q-1)}$ may be much smaller.[9]

In the special case when $\mathbb{F}$ is a *prime field*, this problem can be overcome by starting not from a lower bound for hitting all degree-d_0 polynomials, but rather from a lower bound for hitting a large subcode of the corresponding Reed-Muller code (i.e., a subcode with dimension linear in $\binom{m+d_0}{d_0}$) that still has distance $\Omega(1)$; see [15, Appendix B] for details. To overcome the problem also over non-prime fields, we show a general method that, regardless of the disperser, *allows us to "pay" only an $O(t)$ factor in the degree blow-up*, instead of the 2^ℓ factor. This method does not prevent the additive loss of ℓ in the seed length, and we will explain how this additive loss affects us in the end of the current section.

[9] To demonstrate the problem, note that over fields of constant size, even a disperser with optimal parameters would yield a quadratic degree blow-up, regardless of t; that is, $d \geq 2^\ell \cdot d_0 \geq 2^{\log(t\cdot\log(q)/\delta)} \cdot d_0 = \Omega_q((d_0)^2 \cdot t)$, compared to the previous blow-up of $d = \Omega_q(d_0 \cdot t)$ when we had $\delta = \Omega(1)$.

To explain this method, fix a disperser, and recall that our goal is to "hit" the set $\mathrm{G} \subseteq \mathbb{F}^n$ of inputs z such that for some $i \in \{0,1\}^\ell$ it holds that $p_0(\mathsf{Disp}(z,i)) \neq 0$ (since any $z \in \mathrm{G}$ maps to 2^ℓ inputs, one of which "hits" the original polynomial p_0). We think of the polynomial p above as a test of its input $z \in \mathbb{F}^n$ that distinguishes between G and $\mathbb{F}^n \setminus \mathrm{G}$ (i.e., p vanishes precisely on $\mathbb{F}^n \setminus \mathrm{G}$). Our initial approach to hit G was to construct a HSG for the test p, which would output some $z \in \mathrm{G}$.

The key observation is that constructing a HSG for p is an "overkill". Specifically, to hit G, *we can replace the test p by a distribution $\mathbf{p}$ over tests that distinguishes between* G *and* $\mathbb{F}^n \setminus \mathrm{G}$*, with high probability*, and still deduce that any HSG for the tests in the support of $\mathbf{p}$ outputs some $z \in \mathrm{G}$. That is, we replace the test p for G by a $\mathsf{randomized\ test}$ $\mathbf{p}$ for G such that the polynomials in the support of $\mathbf{p}$ have lower degree than p, and show that "hitting" the polynomials in the support of $\mathbf{p}$ still allows us to "hit" G. Moreover, since $\mathbf{p}$ "tests" a *dense* set G with small error, by an averaging argument almost all of the polynomials in the support of $\mathbf{p}$ *vanish rarely*; thus, it suffices to "hit" only the polynomials in the support of $\mathbf{p}$ that vanish rarely.

More accurately, let us instantiate our disperser with $k = (n-2t) \cdot \log(q)$, instead of $k = (n-t) \cdot \log(q)$, such that the density of G is $1 - q^{-2t}$ (this is to allow for some slackness in the parameters). Then, the following holds:

▶ **Lemma 6** (informal; see Appendix A)**.** *Assume that there exists a distribution $\boldsymbol{p}$ over polynomials $\mathbb{F}^n \to \mathbb{F}$ such that for every $z \in \mathrm{G}$ it holds that $\Pr[\boldsymbol{p}(z) \neq 0] \geq 1 - q^{-2t}$ and for every $z \notin \mathrm{G}$ it holds that $\Pr[\boldsymbol{p}(z) = 0] = 1$. Further assume that every polynomial in the support of $\boldsymbol{p}$ has degree $O(d \cdot t)$. Then, any hitting-set for polynomials of degree $O(d \cdot t)$ that vanish on at most $2q^{-t}$ of their inputs contains some $z \in \mathrm{G}$.*

Our construction of the specific distribution $\mathbf{p}$ that we use is simple: Starting from the construction of p above, instead of taking an $\mathtt{OR}$ of the evaluations of p_0 on the entire output-set of the disperser (i.e., on all seeds), we *sample from the seeds of the disperser*. More accurately, to sample a polynomial $f \sim \mathbf{p}$, we uniformly sample $2t$ vectors $a^{(1)}, ..., a^{(2t)} \in \mathbb{F}^{2^\ell}$, and output the polynomial

$$f(z) = \mathtt{OR}_{j \in [2t]} \left(\sum_{i \in 2^\ell} a_i^{(j)} \cdot p_0(\mathsf{Disp}(z,i)) \right).$$

To see why this distribution works, observe that if $z \in \mathrm{G}$ then a random $\mathbb{F}$-linear sum of the elements $\{\mathsf{Disp}(x,i)\}_{i \in \{0,1\}^\ell}$ will be non-zero with probability $1 - 1/q$, whereas if $z \notin \mathrm{G}$ then such a sum will be zero, with probability one. Thus, a random polynomial in $\mathbf{p}$ computes the disjunction of $2t$ such random sums, and it is straightforward to see that its "error probability" is q^{-2t} and its degree is $O(d_0 \cdot t)$ (assuming that the disperser is linear). Using Lemma 6, any HSG for polynomials of degree $O(d_0 \cdot t)$ that vanish on at most q^{-2t} of their inputs outputs some $z \in \mathrm{G}$. We therefore reduced the problem of constructing a HSG for p_0 to the problem of constructing a HSG for polynomials of degree $d = O(d_0 \cdot t)$ that vanish on at most q^{-2t} of their inputs.

The last missing piece is that we need a concrete disperser to instantiate the argument with, and the parameters of the disperser will determine the lower bound that we get. Furthermore, recall that we are losing an additive factor of ℓ in the lower bound, and thus any lower bound that we get using this approach applies only to values of t such that $\ell \ll d_0 \cdot \log(m/d_0)$. Specifically, the approach above gives the following lemma (for simplicity, we state it only for linear dispersers):

▶ **Lemma 7** (linear dispersers yield lower bounds on HSGs for polynomials that vanish rarely; informal, see Corollary 20). *Let $d_0 < m$ be integers, let $\mathbb{F}$ be a field of size q, and let $t \in \mathbb{N}$. Assume that for $k = (n-2t)\cdot\log(q)$ and $\delta = q^{-d_0/(q-1)}$ there exists a linear (k,δ)-disperser $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$. Then, for $d = 4d_0 \cdot t$, if $\ell \leq \frac{d}{8t}\cdot\log(mt/d)$, then the seed length for any HSG for $\mathcal{P}_{n,q,d,2q^{-t}}$ is $\Omega\left((d/t)\cdot\log\left(mt/d\right)\right)$.*

Note that to get a good lower bound using Lemma 7 we want a *linear* disperser $\mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ for large min-entropy $k = (n-2t)\cdot\log(q)$ that has small seed length ℓ and large output length m.[10] In particular, if there exists a *linear* disperser with *optimal* parameters, then a lower bound of $\Omega((d/t)\cdot\log(nt/d))$ would follow for essentially all settings of the parameters (see Corollary 21).

Our lower bounds, which include Theorem 1 and various extensions and are presented in Section 5, are proved by instantiating Lemma 7 with specific useful dispersers. In a gist, Theorem 1 and some extensions are proved using a linear disperser that we obtain by modifying the extractor by Shaltiel and Umans [36]; the original extractor works over the binary alphabet, and we modify it to a linear disperser over an arbitrary field $\mathbb{F}_q$ (see Appendix B for details). Another extension of Theorem 1, which applies only to fields of constant size, is proved using a linear disperser that is based on the recent construction of "linear 1-local expanders" by Goldreich [17], following Viola and Wigderson [44].

2.2 Explicit Upper Bound Over $\mathbb{F}_2$

To construct the explicit HSG for polynomials $\mathbb{F}_2^n \to \mathbb{F}_2$ that vanish rarely in Theorem 3 we generalize a construction of [18], by extending a proof approach from [40]. In high-level, we reduce the problem of constructing a HSG for polynomials that vanish rarely to the problem of constructing a PRG for arbitrary *low-degree polynomials*, and then use the explicit PRG of Viola [43] for low-degree polynomials.

In more detail, we say that a polynomial $p\colon \mathbb{F}_2^n \to \mathbb{F}_2$ is approximated by a distribution $\mathbf{h}$ over polynomials $h\colon \mathbb{F}_2^n \to \mathbb{F}_2$ if for every $x \in \mathbb{F}_2^n$ it holds that $\Pr_h[\mathbf{h}(x) = p(x)] \geq .99$. Our first step is to show that any polynomial $p \in \mathcal{P}_{n,2,d,q^{-t}}$ can be approximated by a distribution $\mathbf{h}$ over polynomials of degree $d-t$. To do so, let $\Delta_a(p)$ be the directional derivative of p in direction $a \in \mathbb{F}_2^n$ (i.e., the function $\Delta_a p(x) = p(x+a)+p(x)$). We sample $h \sim \mathbf{h}$ by uniformly sampling $\vec{a} = a^{(1)},\ldots,a^{(k)} \in \mathbb{F}_2^n$, where $k = t - O(1)$, and outputting the polynomial $h_{\vec{a}} = \Delta_{a^{(k)}}\Delta_{a^{(k-1)}}\ldots\Delta_{a^{(1)}}(p) + 1$; that is, we derive p in k random directions, and "negate" the output.

Note that indeed $\deg(h_{\vec{a}}) = d - t + O(1)$. Now, for any fixed $x \in \mathbb{F}_2^n$ and non-empty $S \subseteq [k]$, the probability over $\vec{a}$ that $p\left(x + \sum_{i\in S} a^{(i)}\right) = 1$ is at least $1 - 2^{-t}$ (since p vanishes with probability at most 2^{-t}, and $x + \sum_{i\in S} a^{(i)}$ is uniform in $\mathbb{F}_2^n$). Thus, by a union bound, with probability at least .99 over the choice of $\vec{a}$, for every non-empty $S \subseteq [k]$ it holds that $p\left(x + \sum_{i\in S} a^{(i)}\right) = 1$. In this case, we have that $h_{\vec{a}}(x) = \sum_{S\subseteq[k]} p\left(x + \sum_{i\in S} a^{(i)}\right) + 1 = p(x) + (2^k - 1) + 1 = p(x)$. Hence, the distribution $\mathbf{h}$ also has the property that for every $x \in \mathbb{F}_2^n$ it holds that $\Pr[\mathbf{h}(x) = p(x)] \geq .99$.

Our next observation is similar to the "randomized tests" technique mentioned in Section 2.1: We show that if a distribution $\mathbf{h}$ over low-degree polynomials approximates p, then a pseudorandom generator for the polynomials in the support of $\mathbf{h}$ (with sufficiently small constant error) also "hits" p. Combining the two claims, we get a reduction from the

[10] Moreover, since our error $\delta = q^{-d_0/(q-1)}$ might be large, we want good dependency of the parameters ℓ and m on the error δ.

problem of constructing a HSG for $\mathcal{P}_{n,2,d,q^{-t}}$ to the problem of constructing a PRG (with small constant error) for arbitrary polynomials of degree $d-t+O(1)$. Thus, the PRG of Viola [43] for such polynomials, which uses a seed of length $O((d-t)\cdot(2^{d-t}+\log(n)))$, is also a HSG for $\mathcal{P}_{n,2,d,2^{-t}}$.

The proofs of Theorem 2 and Theorem 3 appear in the full version (see [15, Section 5]).

On the tightness of the reduction above

Recall that there is a gap between the seed length of the explicit HSG above and the seed length of the *non-explicit* HSG from Theorem 2, which is $O\left((d-t)\cdot\log(\frac{n}{d-t})\right)$. We note that to close this gap, one does not need to improve the *reduction* detailed above, but only the *explicit PRG for arbitrary polynomials* (i.e., Viola's construction). Specifically, if there exists an explicit PRG for all polynomials of degree $d'=d-t+O(1)$ with seed length $O(d'\cdot\log(n/d'))$ (matching the non-explicit upper-bound for such PRGs), then the reduction above yields a HSG for $\mathcal{P}_{n,2,d,2^{-t}}$ with seed length $O((d-t)\cdot\log(n/(d-t)))$.

3 Preliminaries

We denote random variables by boldface. For an alphabet Σ and $n\in\mathbb{N}$, we denote the uniform distribution over Σ^n by $\mathbf{u}_n$, where Σ will be clear from context.

3.1 Polynomials Over Finite Fields

We consider multivariate polynomials over a finite field. A polynomial $p\colon\mathbb{F}^n\to\mathbb{F}$ of degree d can be viewed as a codeword in the corresponding Reed-Muller code; thus, if p is non-zero, then the relative distance of the corresponding Reed-Muller code, which is stated below, lower bounds the fraction of inputs on which p does not vanish.

▶ **Theorem 8** (distance of the Reed-Muller code; see, e.g., [20]). *For any $d,q\in\mathbb{N}$, let $a=\lfloor d/(q-1)\rfloor$ and $b=d \pmod{q-1}$. The relative distance of the Reed-Muller code of degree d over alphabet q is $\delta_{RM}(d,q)=q^{-a}\cdot(1-b/q)\geq q^{-d/(q-1)}$.*

The $\mathtt{OR}\colon\mathbb{F}^k\to\mathbb{F}$ function maps any non-zero input $z\in\mathbb{F}^k\setminus\{0^k\}$ to $1\in\mathbb{F}$, and maps 0^k to zero. We consider a generalization of this function, which we call *multivalued OR*; a multivalued OR function maps any non-zero $z\in\mathbb{F}^k\setminus\{0^k\}$ to *some* non-zero element (i.e., different non-zero inputs may yield different outputs), while still mapping 0^k to zero. That is:

▶ **Definition 9** (multivalued OR functions). *For any finite field $\mathbb{F}$, we say that a polynomial $\mathtt{mvOR}\colon\mathbb{F}^k\to\mathbb{F}$ is a multivalued OR function if $\mathtt{mvOR}(0^k)=0$, but $\mathtt{mvOR}(x)\neq 0$ for every $x\neq 0^k$.*

For a fixed field $\mathbb{F}$ there are many different k-variate multivalued OR functions. Indeed, the standard $\mathtt{OR}$ function is a multivalued OR function, but it has maximal degree $k\cdot(q-1)$ as a polynomial. We will need k-variate multivalued OR functions that are of much lower degree (i.e., degree approximately k); such functions can be constructed relying on well-known techniques in algebraic geometry (see [40, Proposition 7.3] for the construction, and see e.g. [13, Exercise 8] for a reference to the well-known underlying techniques):

▶ **Proposition 10** (low-degree multivalued OR function). *Let $\mathbb{F}$ be a finite field and let $k\in\mathbb{N}$. Then, there exists a multivalued OR function $\mathtt{mvOR}\colon\mathbb{F}^k\to\mathbb{F}$ that is computable by a polynomial of degree less than $2k$.*

3.2 Hitting-Set Generators

We recall the standard definitions of hitting-set generators (HSGs), of hitting-set generators and of pseudorandom generators (PRGs). Recall that HSGs for a class of polynomials need to produce a set of inputs such that any polynomial from the class evaluates to *non-zero* on some input in the set. That is:

▶ **Definition 11** (hitting-set generator). *Fix a field $\mathbb{F}$, and let $d, n \in \mathbb{N}$. A function $H\colon \{0,1\}^\ell \to \mathbb{F}^n$ is a hitting-set generator for a set of functions $\mathcal{P} \subseteq \{\mathbb{F}^n \to \mathbb{F}\}$ if for every non-zero function $p \in \mathcal{P}$ there exists $s \in \{0,1\}^\ell$ satisfying $p(H(s)) \neq 0$. In this case, the set $S = \{H(s) : s \in \{0,1\}^\ell\}$ is called a hitting-set for $\mathcal{P}$.*

▶ **Definition 12** (explicit hitting-set generators). *Let $\ell, q, d\colon \mathbb{N} \to \mathbb{N}$, let $\{\mathbb{F}_{q(n)}\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ it holds that $\mathbb{F}_{q(n)}$ is a field of size $q(n)$, and let $H = \{H_n\colon \{0,1\}^{\ell(n)} \to \mathbb{F}_{q(n)}^n\}$ such that for every $n \in \mathbb{N}$ it holds that H_n is a hitting-set generator for polynomials of degree $d(n)$. We say that H is polynomial-time computable if there exists an algorithm that gets as input $s \in \{0,1\}^\ell$ and outputs $H_n(s)$ in time $\mathrm{poly}(\ell, \log(q), n)$.*

The standard definition of PRGs for polynomials in $p\colon \mathbb{F}^n \to \mathbb{F}$ that we will use is as follows. Consider the distribution over $\mathbb{F}$ that is obtained by uniformly choosing $x \in \mathbb{F}^n$ and outputting $p(x)$, and the distribution over $\mathbb{F}$ that is obtained by choosing a seed s for a PRG G and outputting $p(G(s))$. We require that the statistical distance between the two distributions is small. That is:

▶ **Definition 13** (pseudorandom generator). *Fix a field $\mathbb{F}$, let $d, n \in \mathbb{N}$, and let $\rho > 0$. A function $G\colon \{0,1\}^\ell \to \mathbb{F}^n$ is a pseudorandom generator with error ρ for polynomials of degree d if for every polynomial $p\colon \mathbb{F}^n \to \mathbb{F}$ of degree at most d it holds that*

$$\sum_{\sigma\in\mathbb{F}} \left| \Pr_{s\in\{0,1\}^\ell}[p(G(s)) = \sigma] - \Pr_{x\in\mathbb{F}^n}[p(x) = \sigma] \right| \leq \rho.$$

An alternative standard definition of PRGs for polynomials requires that the "character distance" $\left|\mathbb{E}_{x\in\mathbb{F}^n}[\mathsf{e}^{p(x)}] - \mathbb{E}_x[\mathsf{e}^{p(G(s))}]\right|$ will be small, where e is any (fixed, non-trivial) character of $\mathbb{F}$. The "character distance" and the statistical distance are equivalent, up to a multiplicative factor of $\sqrt{q-1}$ (see [27, Lemma 2.4]).

Lastly, we recall the standard lower bound on the size of hitting-sets for polynomials of degree d and state the complementary upper-bound that is obtained by a standard probabilistic argument. (For proofs see [15, Section 3].)

▶ **Fact 14** (lower bound on the size of hitting-sets for linear subspaces). *Let $\mathbb{F}$ be a finite field, let $n \in \mathbb{N}$, and let $\mathcal{C} \subseteq \{\mathbb{F}^n \to \mathbb{F}\}$ be a linear subspace of dimension $D = \dim(\mathcal{C})$. Then, any hitting-set for $\mathcal{C}$ has at least D points. In particular, for any $d < n$, any hitting-set for degree-d polynomials $\mathbb{F}^n \to \mathbb{F}$ has size at least $\binom{n+d}{d}$, and correspondingly the seed length of any hitting-set generator for such polynomials is at least $d \cdot \log(n/d)$.*

▶ **Fact 15** (upper bound on the size of hitting-sets). *Let $\mathbb{F}$ be a finite field, let $n \in \mathbb{N}$, and let $d < n$. Then, there exists a (non-explicit) hitting-set generator for polynomials $\mathbb{F}^n \to \mathbb{F}$ of degree d with seed length $O(d \cdot \log(n/d) + \log\log(q))$.*

3.3 Dispersers and Extractors

We recall the definition of dispersers $\mathsf{Disp}\colon [N] \times \{0,1\}^\ell \to [M]$, where we identify the domain N with the vector space $\mathbb{F}^n$ and the range M with the vector space $\mathbb{F}^m$.

▶ **Definition 16** (disperser). *Let $\mathbb{F}$ be a finite field of size $q = |\mathbb{F}|$. A function* $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ *is a (k,δ)-disperser if for every $T \subseteq \mathbb{F}^m$ of size $|T| \geq \delta \cdot q^m$, the probability over $x \in \mathbb{F}^n$ that for all $i \in \{0,1\}^\ell$ it holds that $\mathsf{Disp}(x,i) \notin T$ is less than $2^k/q^n$. The value ℓ is the seed length of the disperser.*[11]

In this work we are interested in dispersers that can be computed by low-degree polynomials. Specifically, we require that for each fixed seed $s \in \{0,1\}^\ell$ and output index $i \in [m]$, the function that maps any $z \in \mathbb{F}^n$ to the i^{th} output of Disp at z with seed s (i.e., $z \mapsto \mathsf{Disp}(z,s)_i$) has low degree as a polynomial $\mathbb{F}^n \to \mathbb{F}$.

▶ **Definition 17** (degree of a disperser). *We say that a disperser* $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ *has degree d if for every fixed $s \in \{0,1\}^\ell$ and $i \in [m]$, the polynomial $p_{s,i}\colon \mathbb{F}^n \to \mathbb{F}$ defined by $p_{s,i}(z) = \mathsf{Disp}(z,s)_i$ is of degree at most d. If $d = 1$, then we say the disperser is linear.*

Recall that there are two standard dispersers that are linear: The naive disperser, which treats its input $z \in \mathbb{F}^n$ as a list of samples from $\mathbb{F}^m$ and its seed as an index of a sample in this list; and the subspace sampler, which treats its input as the description of an affine subspace in $\mathbb{F}^m$ and its seed as an index of an element in the subspace. Nevertheless, these dispersers have disadvantages (small output length and large seed length, respectively), and in our results we will use more sophisticated linear dispersers (see Section 5 for details).

Alternatively, one can verify that Definition 16 is equivalent to the following definition: Disp is a (k,δ)-disperser if for any random variable $\mathbf{x} \sim \mathbb{F}^n$ with min-entropy[12] k, the support of $\mathsf{Disp}(\mathbf{x}, \mathbf{u}_\ell)$ covers at least $(1-\delta)q^m$ elements from $\mathbb{F}^m$. Although dispersers will be our main pseudorandom object, we will sometimes work with the stronger notion of an *extractor*. While in dispersers we only care about covering almost all of $\mathbb{F}^m$, in extractors we want to do it *uniformly*, i.e., we require $\mathsf{Ext}(\mathbf{x}, \mathbf{u}_\ell)$ to be δ-close to the uniform distribution $\mathbf{u}_m$ over $\mathbb{F}^m$. Formally:

▶ **Definition 18** (extractor). *Let $\mathbb{F}$ be a finite field of size $q = |\mathbb{F}|$. A function* $\mathsf{Ext}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ *is a (k,δ)-extractor if for every random variable $\boldsymbol{x} \sim \mathbb{F}^n$ with min-entropy k it holds that $\mathsf{Ext}(\boldsymbol{x}, \boldsymbol{u}_\ell)$ is δ-close to $\boldsymbol{u}_m$. The value ℓ is the seed length of the extractor.*

As the support size of a distribution which is δ-close to $\mathbf{u}_m$ is at least $(1-\delta)q^m$, any (k,δ)-extractor is readily a (k,δ)-disperser.

4 Lower Bounds from Low-Degree Dispersers

In this section we prove general results that use low-degree dispersers to reduce hitting arbitrary polynomials to hitting polynomials that vanish rarely (and thus deduce lower bounds for the latter); this follows the high-level explanations that were presented in Section 2.1. The following proposition specifies the reduction itself, and the subsequent corollary specifies the lower bounds that we can obtain using the reduction.

▶ **Proposition 19** (reducing hitting polynomials to hitting polynomials that vanish rarely by sampling from the seeds of a disperser). *Let $m, d_0 \in \mathbb{N}$, let $\mathbb{F}$ be a field of size q, and let $\delta = \delta_{RM}(d_0, q)$. For $k < \log(q^n)$, let $\varepsilon = 2^k/q^n$, let $\rho < 1-\varepsilon$, and let $r = \log_q(1/\rho)$. Assume that:*

[11] In this work we take the *hitter* view of a disperser, which is equivalent to the following standard definition of dispersers: For every random variable $\mathbf{x} \sim \mathbb{F}^n$ with min-entropy k, $\mathsf{Disp}(\mathbf{x}, \mathbf{u}_\ell)$ has support size at least $(1-\delta)q^m$.

[12] A random variable $\mathbf{x}$ has min-entropy k if for every $x \in \mathsf{supp}(\mathbf{x})$ is holds that $\Pr[\mathbf{x} = x] \leq 2^{-k}$.

1. *There exists a* (k,δ)*-disperser* $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$ *of degree* $d_{\mathsf{Disp}} \in \mathbb{N}$.
2. *There exists a hitting-set* $W \subseteq \mathbb{F}^n$ *for polynomials* $\mathbb{F}^n \to \mathbb{F}$ *of degree* $d = 2d_0 \cdot r \cdot d_{\mathsf{Disp}}$ *that vanish on at most* $\sqrt{\rho+\varepsilon}$ *of their inputs.*

Then, there exists a hitting-set $W_0 \subseteq \mathbb{F}^m$ *for polynomials* $\mathbb{F}^m \to \mathbb{F}$ *of degree* d_0 *such that* $|W_0| \leq |W| \cdot 2^\ell$.

Proof. For $L = 2^\ell$, let $W_0 = \{\mathsf{Disp}(z,i) : z \in W, i \in [L]\}$. We will prove that W_0 is a hitting-set for polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree d_0.

To do so, fix any non-zero polynomial $f\colon \mathbb{F}^m \to \mathbb{F}$ of degree d_0. Let $V = \{x \in \mathbb{F}^m : f(x) = 0\}$ be the set of points on which f vanishes, and let $\mathrm{G} = \{z \in \mathbb{F}^n : \exists i \in [L], \mathsf{Disp}(z,i) \notin V\}$ be the set of inputs $z \in \mathbb{F}^n$ for Disp such that for some $i \in [L]$ it holds that f does not vanish on $\mathsf{Disp}(z,i)$. Note that G has density at least $1-\varepsilon$; this is the case since $|V|/q^m \leq 1-\delta$ (and recall that δ is the distance of the corresponding Reed-Muller code and f is non-zero), and since Disp is a (k,δ)-disperser.

Note that W_0 is a hitting-set for f if and only if $\Pr_{z\in W}[z \in \mathrm{G}] > 0$. We will prove that $\Pr_{z\in W}[z \in \mathrm{G}] > 0$ using Lemma 22. To construct the distribution $\mathbf{p}$ over polynomials in $\mathbb{F}^n \to \mathbb{F}$ needed for the hypothesis of the lemma, fix a multivalued OR polynomial $\mathtt{mvOR}\colon \mathbb{F}^r \to \mathbb{F}$ of degree less than $2r$ as in Proposition 10. Then, sampling $p \sim \mathbf{p}$ is equivalent to the following random process:

> Uniformly and independently choose $\alpha^{(1)}, ..., \alpha^{(r)} \in \mathbb{F}^L$, and output the polynomial $p(z) = \mathtt{mvOR}\left(\sum_{i\in[L]} \alpha_i^{(1)} \cdot f(\mathsf{Disp}(z,i)), ..., \sum_{i\in[L]} \alpha_i^{(r)} \cdot f(\mathsf{Disp}(z,i))\right)$.

Note that each $p \sim \mathbf{p}$ has degree less than $d = d_{\mathsf{Disp}} \cdot d_0 \cdot 2r$. Also note that for any $z \notin \mathrm{G}$ we have that $\Pr[\mathbf{p}(z) = 0] = 1$, whereas for any $z \in \mathrm{G}$ we have that $\Pr[\mathbf{p}(z) \neq 0] \geq 1 - q^{-r} = 1 - \rho$. Using Lemma 22 and the hypothesis that W is a hitting-set for polynomials that vanish on at most $\sqrt{\rho+\varepsilon}$ of their inputs, we deduce that $\Pr_{z\in W}[z \in \mathrm{G}] > 0$, as we wanted. ◀

Using the reduction from Proposition 19, and relying on the unconditional lower bound from Fact 14, we obtain the following result, which uses low-degree dispersers to deduce lower bounds on HSGs for polynomials that vanish rarely:

▶ **Corollary 20** (a lower bound by sampling from the seeds of a disperser)**.** *Let* $m, d_0 \in \mathbb{N}$ *such that* $d_0 < m$*, let* $\mathbb{F}$ *be a field of size* q*, and let* $\delta = \delta_{RM}(d_0, q)$*. For* $t \in \mathbb{N}$ *and* $k = (n-2t)\cdot\log(q)$*, assume that there exists a linear* (k,δ)*-disperser* $\mathsf{Disp}\colon \mathbb{F}^n \times \{0,1\}^\ell \to \mathbb{F}^m$*. Then, any hitting-set* $W \subseteq \mathbb{F}^n$ *for polynomials in* $\mathbb{F}^n \to \mathbb{F}$ *of degree* $d = 4d_0 \cdot t$ *that vanish on at most* $\sqrt{2} \cdot q^{-t}$ *of their inputs has size at least* $\binom{m+d_0}{d_0} \cdot 2^{-\ell}$*. In particular, the seed length for any such hitting-set is at least*

$$\Omega\left(\frac{d}{t} \cdot \log\left(\frac{m \cdot t}{d}\right)\right),$$

provided that $t \leq \frac{\log(mt/d)}{8\ell} \cdot d$.

Proof. We use Proposition 19 with the parameter values $\varepsilon = \rho = q^{-2t} \leq 1/4$ (such that $r = 2t$) and $d_{\mathsf{Disp}} = 1$, and rely on the fact that any hitting-set $W_0 \subseteq \mathbb{F}^m$ for all polynomials $\mathbb{F}^m \to \mathbb{F}$ of degree d_0 has size at least $\binom{m+d_0}{d_0}$ (i.e., on Fact 14). The seed length (in bits) for sampling from the hitting-set is thus at least $d_0 \cdot \log(m/d_0) - \ell = \frac{d}{4t} \cdot \log(4mt/d) - \ell \geq \Omega((d/t) \cdot \log(mt/d)$, where the last inequality is due to the hypothesis that $\frac{d}{4t} \cdot \log(mt/d) \geq 2\ell$. ◀

Finally, note that if there exists a linear (k,δ)-disperser $\mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ with optimal parameters, then we get a lower bound of $\Omega((d/t)\cdot\log(nt/d))$ for essentially all settings of the parameters. That is:

▶ **Corollary 21** (lower bounds assuming an optimal linear disperser)**.** *Assume that for every $n,q,k\in\mathbb{N}$ and $\delta>0$ there exists a* linear (k,δ)*-disperser* $\mathsf{Disp}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ *where $\ell = \log(n\cdot\log(q)-k)+\log(1/\delta)+O(1)$ and $m\cdot\log(q) = k+\ell-\log\log(1/\delta)-O(1)$. Then, for every constant $c>1$ there exists a constant $\gamma>0$ such that the following holds.*

Let $n,q,d,t\in\mathbb{N}$ such that $q\le 2^{n^c}$, and $d<n/2$, and $t\le\gamma\cdot n$, and $\frac{q-1}{\log(q)}\cdot\log(nt/d)\ge 1/\gamma$. Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,\sqrt{2}\cdot q^{-t}}$ is at least $\Omega\left(\frac{d}{t}\cdot\log\left(\frac{n\cdot t}{d}\right)\right)$.

Proof. Let $d_0 = d/4t$, and let $a = d_0/(q-1)$ such that $\delta = \delta_{RM}(d_0,q) \ge q^{-a}$. When instantiating the hypothesized linear disperser with parameters n and $k=(n-2t)\cdot\log(q)$ and $\delta = q^{-a}$, it has seed length $\ell = O(\log(t\cdot\log(q)) + (d/4t)\cdot(\log(q)/(q-1)))$ and output length $m=\Omega(n)$. Relying on Corollary 20, we get a lower bound of $\Omega\left((d/t)\cdot\log(n\cdot(t/d))\right)$, assuming that $d_0<m$ (which holds since we assumed that $d<n/2$) and that $t\le\frac{\log(nt/d)}{8\ell}\cdot d$. Thus, we just need to verify the latter condition.

We verify the condition by a case analysis. The first case is when $t\ge\sqrt{d/4(q-1)}$, which implies that the seed length is $\ell = O(\log(t\cdot\log(q)))$. The condition in this case holds since $\log(nt/d)=\Omega(\log(n))$ and $q\le 2^{\mathrm{poly}(n)}$, which implies that $\frac{\log(nt/d)}{8\ell}=\Omega(1)$. The second case is when $t<\sqrt{d/4(q-1)}$, which implies that the seed length is $\ell = O((d/t)\cdot\log(q)/(q-1))$. The condition in this case holds if and only if $\frac{q-1}{\log(q)}\cdot\log(nt/d)$ is larger than a sufficiently large constant, which is our hypothesis. ◀

5 Lower Bounds Over General Finite Fields

In this section we describe our lower bounds on the seed length of HSGs for polynomials that vanish rarely, which are proved by instantiating the approach from Section 4 with specific dispersers that are suitable for the corresponding parameter settings.

We prove three incomparable lower bounds. Our first and main lower bound is a generalization of Theorem 1. This lower bound is of the form $\Omega((d/t)\cdot\log(n^{1-\Omega(1)}t/d))$, and holds under complicated conditions on the degree d and on t; in particular, for $d\le n^{.49}$ as in Theorem 1, it holds for all values of t up to $\Omega(d)$. For details see Theorem 23 in Appendix B.

Our two additional lower bounds, which are detailed and proved in the full version (see [15, Section 6.3]), hold in more specific settings than the foregoing lower bound, but have advantages over this bound. The first of the two lower bounds holds only when $d\le q$ (i.e., when the corresponding Reed-Muller code has distance $\Omega(1)$); this lower bound is of the same form as in Theorem 23, but holds for higher degrees up to $d\le n^{1-\Omega(1)}$ without complicated conditions on d and t. The second lower bound holds only over fields of constant size; this lower bound is of the stronger form $\Omega((d/t)\cdot\log(nt/d))$,[13] and holds for degrees d up to $\Omega(n)$, but only for value of $t\lessapprox\sqrt{d}$.

References

1 Emmanuel Abbe, Amir Shpilka, and Avi Wigderson. Reed-Muller codes for random erasures and errors. *IEEE Transactions on Information Theory*, 61(10):5229–5252, 2015.

2 N. Alon, J. Bruck, J. Naor, M. Naor, and R. M. Roth. Construction of asymptotically good low-rate error-correcting codes through pseudo-random graphs. *IEEE Transactions on Information Theory*, 38(2):509–516, 1992.

[13] Recall, from Corollary 21, that this is the lower is that would be obtained if there exists a linear disperser with optimal parameters.

3 Peter Beelen and Mrinmoy Datta. Generalized Hamming weights of affine Cartesian codes. *Finite Fields and their Applications*, 51:130–145, 2018.

4 E. R. Berlekamp and N. J. A. Sloane. Restrictions on weight distribution of Reed-Muller codes. *Information and Control*, 14:442–456, 1969.

5 Arnab Bhattacharyya. Polynomial decompositions in polynomial time. In *Proc. 22nd European Symposia on Algorithms*, pages 125–136. Springer, 2014.

6 Arnab Bhattacharyya, Abhishek Bhowmick, and Chetan Gupta. On higher-order Fourier analysis over non-prime fields. In *Proc. 20th International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages Art. No. 23, 29. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.23.

7 Arnab Bhattacharyya, Pooya Hatami, and Madhur Tulsiani. Algorithmic regularity for polynomials and applications. In *Proc. 26th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1870–1889, 2015.

8 Markus Bläser, Moritz Hardt, and David Steurer. Asymptotically optimal hitting sets against polynomials. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming, Part I*, Proc. 35th International Colloquium on Automata, Languages and Programming (ICALP), pages 345–356, 2008.

9 Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *Proc. 37th Annual ACM Symposium on Theory of Computing (STOC)*, pages 21–30. ACM, 2005. doi:10.1145/1060590.1060594.

10 Andrej Bogdanov and Emanuele Viola. Pseudorandom bits for polynomials. *SIAM Journal of Computing*, 39(6):2464–2486, 2010.

11 Lijie Chen and Roei Tell. Bootstrapping results for threshold circuits "just beyond" known lower bounds. In *Proc. 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 34–41, 2019.

12 Gil Cohen and Amnon Ta-Shma. Pseudorandom generators for low degree polynomials from algebraic geometry codes. *Electronic Colloquium on Computational Complexity: ECCC*, 20:155, 2013.

13 David A. Cox, John Little, and Donal O'Shea. *Ideals, varieties, and algorithms*. Undergraduate Texts in Mathematics. Springer, Cham, fourth edition, 2015.

14 Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. *Electronic Colloquium on Computational Complexity: ECCC*, 26:99, 2019.

15 Dean Doron, Amnon Ta-Shma, and Roei Tell. On hitting-set generators for polynomials that vanish rarely (rev. 1). *Electronic Colloquium on Computational Complexity: ECCC*, 26:119, 2019.

16 Zeev Dvir. On the size of Kakeya sets in finite fields. *Journal of the American Mathematical Society*, 22(4):1093–1097, 2009.

17 Oded Goldreich. Deconstructing 1-local expanders. *Electronic Colloquium on Computational Complexity: ECCC*, 23:152, 2016.

18 Oded Goldreich and Avi Widgerson. On derandomizing algorithms that err extremely rarely. In *Proc. 46th Annual ACM Symposium on Theory of Computing (STOC)*, pages 109–118. ACM, 2014. Full version available online at *Electronic Colloquium on Computational Complexity: ECCC*, 20:152 (Rev. 2), 2013. doi:10.1145/2591796.2591808.

19 Ben Green and Terence Tao. The distribution of polynomials over finite fields, with applications to the Gowers norms. *Contributions to Discrete Mathematics*, 4(2):1–36, 2009.

20 Venkatesan Guruswami, Atri Rudra1, and Madhu Sudan. Essential coding theory, 2019. Accessed at https://cse.buffalo.edu/faculty/atri/courses/coding-theory/book/web-coding-book.pdf.

21 Elad Haramaty and Amir Shpilka. On the structure of cubic and quartic polynomials. In *Proc. 42nd Annual ACM Symposium on Theory of Computing (STOC)*, pages 331–340. ACM, 2010. doi:10.1145/1806689.1806736.

22 Tadao Kasami and Nobuki Tokura. On the weight structure of Reed-Muller codes. *IEEE Transactions on Information Theory*, IT-16:752–759, 1970.

23 Tali Kaufman and Shachar Lovett. Worst case to average case reductions for polynomials. In *Proc. 49th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 166–175, 2008.

24 Tali Kaufman, Shachar Lovett, and Ely Porat. Weight distribution and list-decoding size of Reed-Muller codes. *IEEE Transactions on Information Theory*, 58(5):2689–2696, 2012.

25 Adam R. Klivans and Daniel Spielman. Randomness efficient identity testing of multivariate polynomials. In *Proc. 33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 216–223, 2001.

26 Daniel Lewin and Salil Vadhan. Checking polynomial identities over any field: towards a derandomization? In *Proc. 30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 438–447. ACM, 1998. doi:10.1145/276698.276856.

27 Shachar Lovett. Unconditional pseudorandom generators for low-degree polynomials. *Theory of Computing*, 5:69–82, 2009.

28 Chi-Jen Lu. Hitting set generators for sparse polynomials over any finite fields. In *Proc. 27th Annual IEEE Conference on Computational Complexity (CCC)*, pages 280–286. IEEE Computer Society, 2012. doi:10.1109/CCC.2012.20.

29 M. Luby, B. Velickovic, and A. Wigderson. Deterministic approximate counting of depth-2 circuits. In *Proc. 2nd Israel Symposium on Theory and Computing Systems*, pages 18–24, 1993.

30 F. J. MacWilliams and N. J. A. Sloane. *The theory of error-correcting codes. II.* North-Holland Publishing Co., Amsterdam-New York-Oxford, 1977.

31 R. J. McEliece. Quadratic forms over finite fields and second-order Reed-Muller codes. *Space Program Summary*, 3(37–58):28–33, 1969.

32 Joseph Naor and Moni Naor. Small-bias probability spaces: efficient constructions and applications. *SIAM Journal of Computing*, 22(4):838–856, 1993.

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.

35 Rocco A. Servedio and Li-Yang Tan. Luby-Veličković-Wigderson revisited: improved correlation bounds and pseudorandom generators for depth-two circuits. In *Proc. 22nd International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, volume 116, pages Art. No. 56, 20. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.APPROX-RANDOM.2018.56.

36 Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *Journal of the ACM*, 52(2):172–216, 2005.

37 Neil J. A. Sloane and Elwyn R. Berlekamp. Weight enumerator for second-order Reed-Muller codes. *IEEE Transactions on Information Theory*, IT-16:745–751, 1970.

38 Amnon Ta-Shma and David Zuckerman. Extractor codes. *IEEE Transactions on Information Theory*, 50(12):3015–3025, 2004.

39 Amnon Ta-Shma, David Zuckerman, and Shmuel Safra. Extractors from Reed-Muller codes. *Journal of Computer and System Sciences*, 72(5):786–812, 2006.

40 Roei Tell. Improved bounds for quantified derandomization of constant-depth circuits and polynomials. In *Computational Complexity*, 2019.

41 Salil P. Vadhan. *Pseudorandomness.* Foundations and Trends in Theoretical Computer Science. Now Publishers, 2012.

42 Emanuele Viola. Guest column: correlation bounds for polynomials over $\{0,1\}$. *SIGACT News*, 40:27–44, February 2009.

43 Emanuele Viola. The sum of d small-bias generators fools polynomials of degree d. *Computational Complexity*, 18(2):209–217, 2009.

44 Emanuele Viola and Avi Wigderson. Local expanders. *Computational Complexity*, 2017.

45 Ewald Warning. Bemerkung zur vorstehenden arbeit von herrn chevalley. *Abhandlungen aus dem Mathematischen Seminar der Universität Hamburg*, 1935.

A Randomized Tests

The proofs of both our upper bounds and of our lower bounds relies on a general observation that we explain here. The observation is essentially from [40, Sections 2.1 & 4], following a proof idea from [10].

Assume that we want to deterministically *find* an element in a set $\mathrm{G} \subseteq \mathbb{F}^n$. A standard way to do so is to show that G can by decided by a simple algorithm p (e.g., p is a low-degree polynomial), which we think of as a *simple test*. Then, a hitting-set generator for p outputs an element in G. Our goal now is to find an element in G using a hitting-set generator for tests that are *simpler* than p. The basic observation is that if G can be decided, with high probability, by a *distribution* $\boldsymbol{p}$ *over simple tests*, then a hitting-set generator with small density for the tests in the *support* of $\mathbf{p}$ outputs an element in G (see [40, Observation 2.1]). The advantage is that instead of constructing a deterministic test p we can now construct a *randomized test* $\boldsymbol{p}$, whose complexity is potentially lower than that of p; that is, the complexity of the tests in the support of the distribution $\mathbf{p}$ may be lower than the complexity of the deterministic test p.

The observation above can be extended in various ways (see [40] for details), and we will apply it in two specific settings. In the first setting, which is useful for our lower bound proofs, the set G is dense (i.e., $\Pr_{x\in\mathbb{F}^n}[x \in \mathrm{G}] \geq .99$), and can be decided by a distribution $\mathbf{p}$ over polynomials with small "one-sided" error (i.e., every $x \in \mathrm{G}$ is accepted with high probability, and every $x \notin \mathrm{G}$ is rejected with probability one). We show that in this case, any hitting-set generator for the polynomials in the support of $\mathbf{p}$ that *vanish rarely* outputs an element in G (and this holds without any density requirement from the HSG).

▶ **Lemma 22** (randomized tests). *Let $\varepsilon, \rho > 0$ such that $\varepsilon + \rho < 1$, and let $\mathrm{G} \subseteq \mathbb{F}^n$ be such that $\Pr_{x\in\mathbb{F}^n}[x \in \mathrm{G}] \geq 1 - \varepsilon$. Assume that there exists a distribution $\boldsymbol{p}$ over polynomials $p\colon \mathbb{F}^n \to \mathbb{F}$ such that:*

1. *For every fixed $x \in \mathrm{G}$ it holds that $\Pr[\boldsymbol{p}(x) \neq 0] \geq 1 - \rho$.*
2. *For every fixed $x \notin \mathrm{G}$ it holds that $\Pr[\boldsymbol{p}(x) = 0] = 1$.*

Let $\boldsymbol{w}$ be a distribution over $\mathbb{F}^n$ such that for every $p\colon \mathbb{F}^n \to \mathbb{F}$ in the support of $\boldsymbol{p}$ that vanishes on at most a $\sqrt{\rho + \varepsilon}$ fraction of its inputs there exists $w \sim \boldsymbol{w}$ such that $p(w) \neq 0$. Then, there exists $w \sim \boldsymbol{w}$ such that $w \in \mathrm{G}$.

We give the proof of Lemma 22 in the full version of the paper (see [15, Section 4]).

In the second setting, which is useful for our upper-bound proof (see [15, Section 5]), we want to "fool" a polynomial $p\colon \mathbb{F}^n \to \mathbb{F}$ using a pseudorandom generator for polynomials that are simpler than p (e.g., they are of lower degree). This is indeed possible if there is a distribution $\mathbf{h}$ over polynomials that are simpler than p such that for every fixed $x \in \mathbb{F}^n \to \mathbb{F}$ it holds that $\Pr[\mathbf{h}(x) = p(x)]$ is high. We defer the details of the second setting to the full version (see [15, Section 4]).

B The Main Lower Bound: Proof of Theorem 1

In this section we prove lower bounds that hold also when the degree is much larger than the field size (i.e., $d \gg q$). Specifically, we will prove the following, more general version of Theorem 1:

▶ **Theorem 23** (a lower bound using the Shaltiel-Umans linear disperser; a more general version of Theorem 1). *For any two constants $\gamma > 0$ and $\gamma' > 0$ there exists a constant $\gamma'' > 0$ such that the following holds. Let $n, d, t, q \in \mathbb{N}$ such that $q \leq n^{1/\gamma'}$ is a prime power, $d \leq n/4$, and:*

- *(essentially all values of $\varepsilon = q^{-t}$) $t \leq \gamma'' \cdot \frac{\log(nt/d)}{\log(n)} \cdot d$.*
- *(auxiliary condition that holds for typical settings) $\frac{q-1}{\log(q)} \cdot \log(nt/d) \geq 1/\gamma''$.*
- *(main condition: d/t is upper-bounded) $d/t \leq \gamma'' \cdot \min\left\{\frac{q-1}{\log(q)} \cdot n^{\gamma}, n^{1-(\gamma+\gamma')}\right\}$.*

Then, the seed length of any HSG for $\mathcal{P}_{n,q,d,\sqrt{2}\cdot q^{-t}}$ is at least $\Omega\left(\frac{d}{t} \cdot \log\left(\frac{n^{1-(\gamma+\gamma')}\cdot t}{d}\right)\right)$.

To deduce Theorem 1 from Theorem 23, note that if we are willing to assume that $d \leq n^{.49}$, then we can choose $\gamma = .499$ and $\gamma' > 0$ that is sufficiently small, and the three conditions in Theorem 23 hold for every $q \leq n^{1/\gamma'}$ and $t \leq \gamma'' \cdot d$.

To prove Theorem 23 we will instantiate Corollary 20 with a linear disperser that we will construct relying on the extractor of Shaltiel and Umans [36]. Recall that [36] constructed an extractor $\mathsf{Ext}\colon \{0,1\}^n \times \{0,1\}^\ell \to \{0,1\}^m$ by first constructing what they called a q-ary extractor, whose output lies in a field of size $\mathrm{poly}(n)$ and only satisfies a relatively-weak unpredictability requirement, and then transforming the q-ary extractor to a standard extractor over the binary alphabet (the transformation follows an idea of Ta-Shma, Zuckerman, and Safra [39]).

We want to construct a low-degree disperser $\mathsf{Disp}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q$ where the field $\mathbb{F}_q$ is of size much smaller than $\mathrm{poly}(n)$ (i.e., $q \leq n^{\gamma'}$ for some small constant $\gamma' > 0$). To do so, we take as a starting-point their construction of a q_0-ary extractor from [36], where $q_0 = \mathrm{poly}(n)$, and then generalize their transformation of q_0-ary extractors to standard extractors (and in particular dispersers) such that the resulting extractor is both over the field $\mathbb{F}_q$, rather than over a binary alphabet, and linear.

Towards presenting the construction, let us first recall the definition of q_0-ary extractors and the main construction of such objects from [36].

▶ **Definition 24** (q_0-ary extractor). *For $n, k, m, \ell \in \mathbb{N}$ and $\rho > 0$, and a prime power $q_0 \in \mathbb{N}$, we say that $\mathsf{Ext}_0\colon \mathbb{F}_{q_0}^n \times \{0,1\}^\ell \to \mathbb{F}_{q_0}^m$ is a (k,ρ) q_0-ary extractor if for every random variable $\boldsymbol{x}$ over $\mathbb{F}_{q_0}^n$ with min-entropy at least k, and every $i \in [m]$, and every function $P\colon \mathbb{F}_{q_0}^{i-1} \to \mathbb{F}_{q_0}^{\rho^{-2}}$, it holds that $\Pr_{x\sim\boldsymbol{x},u\sim\boldsymbol{u}_\ell}[P(\mathsf{Ext}_0(x,u)_1, ..., \mathsf{Ext}_0(x,u)_{i-1}) \ni \mathsf{Ext}_0(x,u)_i] \leq \rho$.*

▶ **Theorem 25** ([36, Theorem 4.5, Item 1]). *There exists a universal constant $c > 1$ such that the following holds. Let $n_0, q_0, k, m, r, h \in \mathbb{N}$ and $\rho > 0$ such that q_0 is a prime power, and the following inequalities hold:*

1. *(sufficiently large auxiliary parameters h and r) $n_0 \leq \binom{h+r-1}{r}$.*
2. *(sufficiently large field) $q_0 \geq c \cdot \frac{(h\cdot r)^2}{\rho^4}$.*
3. *(sufficiently small output length) $m \leq \frac{k - \log(1/\rho)}{c\cdot h\cdot r\cdot \log(q_0)}$.*

Then, there exists an $r \times r$ matrix A over $\mathbb{F}_{q_0}$ such that the following holds. Let $\mathsf{Ext}_0\colon \mathbb{F}_{q_0}^{n_0} \times \{0,1\}^{r\cdot\log(q_0)} \to \mathbb{F}_{q_0}^m$ be defined by $\mathsf{Ext}_0(x,v) = p_x(A^1\cdot v) \circ p_x(A^2\cdot v) \circ ... \circ p_x(A^m \cdot v)$, where v is interpreted as an element in $\mathbb{F}_{q_0}^r$, and $p_x\colon \mathbb{F}_{q_0}^r \to \mathbb{F}_{q_0}$ is the r-variate polynomial of total degree $h-1$ whose coefficients are specified by x. Then, Ext_0 is a (k,ρ) q_0-ary extractor.

Note that in [36] the input of the extractor is represented in binary and interpreted as n_0 elements in $\mathbb{F}_q$, whereas in Theorem 25 we considered the input as n_0 elements in $\mathbb{F}_q$. The two formulations are equivalent, since a random variable over $\mathbb{F}_{q_0}^{n_0}$ has min-entropy k if and only if the corresponding random variable over $\{0,1\}^{n_0 \cdot \log(q_0)}$ has min-entropy k. Also note that [36, Lemma 4.4] showed that A can be constructed in time $q_0^{O(r)}$ (by an exhaustive search over the field $\mathbb{F}_{(q_0)^r}$), and deduced that the extractor is efficiently computable; however, we will not use this property of the extractor.

We now present the transformation of q_0-ary extractors to standard extractors whose inputs and outputs are vectors over $\mathbb{F}_q$, where $q \ll q_0$; as mentioned above, the proof, given in the full version, generalizes an idea from [39]. The intuition for this transformation is the following. Consider the output distribution of a q_0-ary extractor as consisting of blocks of elements from $\mathbb{F}_q$, where each block represents a single element from $\mathbb{F}_{q_0}$; by definition, the output distribution of a q_0-ary extractor is "next-element unpredictable", and hence the distribution of elements from $\mathbb{F}_q$ is a *block source* (see, e.g., [41, Section 6.3.1]). Following Nisan and Zuckerman [34], we compose the q_0-ary extractor with a strong extractor over $\mathbb{F}_q$ that outputs a single element (and maps each block to a single element) and obtain an extractor over $\mathbb{F}_q$. We will specifically use a single-output extractor that is obtained from a *linear list-decodable code* (see, e.g., [38, Claim 4.1]), relying on well-known constructions of such codes.[14]

▶ **Proposition 26** (transforming a q_0-ary extractor into a standard extractor over $\mathbb{F}_q$)**.** *Let $\rho > 0$, let q be a prime power, let $q_0 = q^\Delta$ for some $\Delta \in \mathbb{N}$, and let $\mathfrak{C}\colon \mathbb{F}_q^\Delta \to \mathbb{F}_q^\Delta$ be a $(1 - 1/q - \rho, \rho^{-2})$-list-decodable code. Assume that $\mathsf{Ext}_0\colon \mathbb{F}_{q_0}^{n_0} \times \{0,1\}^{\ell_0} \to \mathbb{F}_{q_0}^m$ is a (k,ρ) q_0-ary extractor. Let $\mathsf{Ext}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$, where $n = n_0 \cdot \Delta$ and $\ell = \ell_0 + \log(\bar{\Delta})$, be defined by*

$$\mathsf{Ext}(x,(y,j)) = \mathfrak{C}(\mathsf{Ext}_0(\hat{x},y)_1)_j \circ \ldots \circ \mathfrak{C}(\mathsf{Ext}_0(\hat{x},y)_m)_j \ ,$$

where $\hat{x} \in \mathbb{F}_{q_0}^{n_0}$ is the vector that is represented by $x \in \mathbb{F}_q^{n_0 \cdot \Delta}$. Then, Ext is a $(k, 2qm \cdot \rho)$-extractor.

We now combine Theorem 25 and Proposition 26 to obtain a linear (k,δ)-disperser $\mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$ with output length $m = k/n^{\Omega(1)}$ and seed length $\ell = O(\log(n/\delta))$. We provide the proof in the full version of the paper.

▶ **Theorem 27** (an adaptation of the Shaltiel-Umans extractor to a linear disperser over general finite fields)**.** *For any two constants $\gamma, \gamma' > 0$ the following holds. Let $n, k, q \in \mathbb{N}$ such that $k \geq n^{\gamma+\gamma'}$ and $q \leq n^{1/\gamma'}$, and let $\delta \geq 2^{-n^\gamma + \log(2qn)}$. Then, there exists a linear (k,δ)-disperser $\mathsf{Disp}\colon \mathbb{F}_q^n \times \{0,1\}^\ell \to \mathbb{F}_q^m$, where $\ell = O_{\gamma'}(\log(n/\delta))$ and $m = \Omega_{\gamma'}\left(k/n^{\gamma+\gamma'}\right)$.*

Finally, we deduce our lower bound from Theorem 23 using Corollary 20 with the linear disperser from Theorem 27.

Proof of Theorem 23. Let $d_0 = d/4t$, and let $a = d_0/(q-1)$ such that $\delta = \delta_{RM}(d_0, q) \geq q^{-a}$. We instantiate the linear disperser from Theorem 27 with parameters n and $k = (n-2t)\cdot\log(q)$ and $\delta = q^{-a} \geq 2^{-n^\gamma + \log(2qn)}$, and with the parameters $\gamma > 0$ and $\gamma' > 0$. The conditions of Theorem 27 hold due to our hypotheses that $d/t \leq \gamma'' \cdot \frac{q-1}{\log(q)} \cdot n^\gamma$ (which implies that

[14] In fact, since in our case the output of the q_0-ary extractor is not only unpredictable but also unpredictable by predictors that output a *list* of elements, we use a simpler proof that does not go through the notion of strong extractors.

$\delta \geq 2^{-n^{\gamma}+\log(2qn)}$) and that $d \leq n/4$ (which implies that $k = \Omega(n)$). For these parameters, the disperser has seed length $\ell = O(\log(n/\delta)) = O(\log(n) + (d/4t) \cdot (\log(q)/(q-1)))$ and output length $m = \Omega(n^{1-(\gamma+\gamma')})$.

Relying on Corollary 20, we get a lower bound of $\Omega\left((d/t) \cdot \log(n^{1-(\gamma+\gamma')} \cdot (t/d))\right)$, assuming that $d_0 < m$ (which holds since $d/4t < \gamma'' \cdot n^{1-(\gamma+\gamma')}$) and that $t \leq \frac{\log(nt/d)}{8\ell} \cdot d$. Thus, we just need to verify the latter condition.

We verify the condition by a case analysis. The first case is when $\log(n) > \frac{d\log(q)}{4t(q-1)}$, which implies that the seed length is $\ell = O(\log(n))$; then, the condition that we want holds due to our hypothesis $t \leq \gamma'' \cdot \frac{\log(nt/d)}{\log(n)} \cdot d$. In the second case we have that $\frac{d\log(q)}{4t(q-1)} \geq \log(n)$, which implies that the seed length is $\ell = O\left(\frac{d\log(q)}{t(q-1)}\right)$; then, the condition holds since we assumed that $\frac{q-1}{\log(q)} \cdot \log(nt/d) \geq 1/\gamma''$. ◀

C Small Sets With a Large Degree-d Closure

In this section we establish a connection between the study of HSGs for polynomials that vanish rarely, and the study of small sets with large degree-d closures, which was recently initiated by Nie and Wang [33]. To do so let us first define the degree-d closure of a set $S \subseteq \mathbb{F}^n$:

▶ **Definition 28** (degree-d closure). *Let $\mathbb{F}$ be a finite field, and let $n, d \in \mathbb{N}$. Then, for any $S \subseteq \mathbb{F}^n$, we define the degree-d closure of S, denoted $\mathtt{Cl}^{(\mathtt{d})}(S)$, by $\mathtt{Cl}^{(\mathtt{d})} = \{x \in \mathbb{F}^n : \forall p \in \mathcal{I}(S), p(x) = 0\}$, where $\mathcal{I}(S) = \{p : \mathbb{F}^n \to \mathbb{F} : \deg(p) = d \wedge \forall s \in S, p(s) = 0\}$.*

We now restate and prove Theorem 5, which shows two reductions. Loosely speaking, we show that any set with degree-d closure of size q^{n-t} is a hitting-set for polynomials that vanish with probability at most q^{-t}; and we show that any hitting-set for polynomials that vanish with probability at most q^{-t} has degree-d' closure of size $q^{n-t}/2$, for d' that is not much smaller than d.

▶ **Theorem 5** (small sets with large closures versus hitting-sets for polynomials that vanish rarely). *Let $\mathbb{F}$ be a field of size q, let $n \in \mathbb{N}$ and $t < d < n$, and let $S \subseteq \mathbb{F}^n$. Then,*

1. *If $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| > q^{n-t}$, then S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$.*
2. *If S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$, then $\left|\mathtt{Cl}^{(\mathtt{d}/2(\mathtt{t}+1))}(S)\right| > \frac{1}{2} \cdot q^{n-t}$.*

Proof. For the first statement, let $S \subseteq \mathbb{F}^n$ be such that $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right| > q^{n-t}$. Then, every degree-$d$ polynomial that vanishes on S also vanishes on more than q^{n-t} of the inputs. It follows that S is a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$.

For the second statement, for $d' = d/2(t+1)$, assuming that $\left|\mathtt{Cl}^{(\mathtt{d}')}(S)\right| \leq \frac{1}{2} \cdot q^{n-t}$, we construct a degree-$d$ polynomial that vanishes on S and that vanishes on at most q^{n-t} inputs in $\mathbb{F}^n$ (and it follows that S is not a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$).

To construct the polynomial, let $T_1 = \mathbb{F}^n \setminus \mathtt{Cl}^{(\mathtt{d}')}(S)$. Note that for every $x \in T_1$ there exists a degree-d' polynomial p_x that vanishes on S, but does not vanish at x. We can thus construct a collection $\mathcal{P}_1$ of degree-d' polynomials such that for every $x \in T_1$ there exists a corresponding $p_x \in \mathcal{P}_1$ satisfying $p_x(x) \neq 0$. (Indeed, a single polynomial might "cover" two distinct inputs, i.e. $p_x = p_y$ for $x \neq y$.)

Now, consider the distribution $\mathbf{p}_1$ over polynomials $\mathbb{F}^n \to \mathbb{F}$ that is defined by

$$\mathbf{p}_1(z) = \sum_{x \in T_1} \mathbf{c}_x \cdot p_x(z),$$

where the coefficients $\mathbf{c}_x$ are uniformly and independently chosen in $\mathbb{F}$. Note that $\mathbf{p}_1$ is supported by polynomials of degree d' that vanish on S. Also note that for any fixed $z \in T_1$ we have that

$$\begin{aligned}\Pr[\mathbf{p}_1(z)=0] &= \Pr\left[\sum_{x\in T_1}\mathbf{c}_x\cdot p_x(z)=0\right] \\ &= \mathop{\mathbb{E}}_{\{c_x\}_{x\in T_1\setminus\{z\}}}\left[\Pr\left[\mathbf{c}_z\cdot p_z(z)=-\sum_{x\in T_1\setminus\{z\}}c_x\cdot p_x(z)\right]\right],\end{aligned}$$

which equals $1/q$ since $p_z(z)\neq 0$. Therefore, there exists a fixed polynomial p of degree d' that vanishes on S and on at most $1/q$ of the inputs in T_1.

We now repeat this step t additional times, while maintaining the invariant that for every $x \in T_i$ there exists a polynomial $p_x \in \mathcal{P}_i$ such that $p_x(x)\neq 0$. Specifically, for $i=2,...,t+1$, we let $T_i = T_{i-1}\cap\{x\in T_i : p_{i-1}(x)=0\}$ and $\mathcal{P}_i = \mathcal{P}_{i-1}\setminus\{p_{i-1}\}$. Note that $|T_i|\leq |T_{i-1}|/q$, and that for every $x\in T_i$ there exists $p_x\in\mathcal{P}_i$ such that $p_x(x)\neq 0$. We again define a distribution $\mathbf{p}_i(z)=\sum_{x\in T_i}\mathbf{c}_x\cdot p_x(z)$, and using the same argument as above, we deduce that there exists a fixed polynomial p_i of degree d' that vanishes on S and on at most $1/q$ of the inputs in T_i.

After $t+1$ steps we obtain $t+1$ polynomials $p_1,...,p_{t+1}$ of degree d' that vanish on S such that $\left|\{x\notin \mathtt{Cl}^{(\mathtt{d})}(S) : \forall i\in[t], p_i(x)=0\}\right| \leq |T_1|/q^{t+1}\leq \frac{1}{2}\cdot q^{-t}$. Let $p\colon \mathbb{F}^n\to\mathbb{F}$ be the multivalued OR of $p_1,...,p_{t+1}$, defined by $p(x)=\mathtt{mvOR}(p_1(x),...,p_t(x))$. Note that $\deg(p)<2(t+1)\cdot d'=d$, and that p vanishes on S. Thus, denoting $\delta=\left|\mathtt{Cl}^{(\mathtt{d}')}(S)\right|/q^n\leq\frac{1}{2}\cdot q^{-t}$, we have that

$$\Pr_{x\in\mathbb{F}^n}[p(x)=0]=\delta+(1-\delta)\cdot q^{-(t+1)}<q^{-t}.$$

which implies that $p\in\mathcal{P}_{n,q,d,q^{-t}}$. Hence, S is not a hitting-set for $\mathcal{P}_{n,q,d,q^{-t}}$. ◀

As mentioned in Section 1.3, we can obtain an upper-bound on the size of $\mathtt{Cl}^{(\mathtt{d})}(S)$ for any sufficiently-small set S, by combining Theorem 23 and the first item of Theorem 5. Specifically, we can deduce that for every $2\leq q\leq \mathrm{poly}(n)$ and $d\leq n^{.49}$ and $t\leq\gamma\cdot d$ (where $\gamma>0$ is a sufficiently small constant), any set S of size $|S|\leq n^{\gamma\cdot(d/t)}$ satisfies $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right|\leq q^{n-t}$. However, this corollary is superseded by the upper-bound of [33], who showed that for any $S\subseteq\mathbb{F}^n$ it holds that $\left|\mathtt{Cl}^{(\mathtt{d})}(S)\right|\leq\frac{|S|}{\binom{n+d}{d}}\cdot q^n$.

Indeed, since the problem of constructing small sets with large degree-d closures is at least as hard as the problem of constructing HSGs for polynomials that vanish rarely (due to the first item of Theorem 5), it might be inherent that a direct lower bound on the former problem is stronger than a lower bound that is obtained via a reduction from the latter problem.

Polynomial Identity Testing for Low Degree Polynomials with Optimal Randomness

Markus Bläser
Department of Computer Science, Saarland University, Saarland Informatics Campus, Saarbrücken, Germany
mblaeser@cs.uni-saarland.de

Anurag Pandey
Max Planck Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
apandey@mpi-inf.mpg.de

Abstract

We give a randomized polynomial time algorithm for polynomial identity testing for the class of n-variate poynomials of degree bounded by d over a field $\mathbb{F}$, in the blackbox setting.

Our algorithm works for every field $\mathbb{F}$ with $|\mathbb{F}| \geq d+1$, and uses only $d \log n + \log(1/\epsilon) + O(d \log \log n)$ random bits to achieve a success probability $1-\epsilon$ for some $\epsilon > 0$. In the low degree regime that is $d \ll n$, it hits the information theoretic lower bound and differs from it only in the lower order terms. Previous best known algorithms achieve the number of random bits (Guruswami-Xing, CCC'14 and Bshouty, ITCS'14) that are constant factor away from our bound. Like Bshouty, we use Sidon sets for our algorithm. However, we use a new construction of Sidon sets to achieve the improved bound.

We also collect two simple constructions of hitting sets with information theoretically optimal size against the class of n-variate, degree d polynomials. Our contribution is that we give new, very simple proofs for both the constructions.

2012 ACM Subject Classification Theory of computation → Algebraic complexity theory

Keywords and phrases Algebraic Complexity theory, Polynomial Identity Testing, Hitting Set, Pseudorandomness

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.8

Category RANDOM

Funding *Anurag Pandey*: Supported by the Chair of Raimund Seidel, Department of Computer Science, Saarland University, Saarbrücken, Germany.

Acknowledgements We thank Rohit Gurjar, Mrinal Kumar and Raimund Seidel for insightful discussions. We thank the Simons Institute for the Theory of Computing (Berkeley) and Schloss Dagstuhl – Leibniz-Zentrum für Informatik (Dagstuhl), for hosting us during certain phases of this research.

1 Introduction

We investigate algorithms for the problem of Polynomial Identity testing (PIT). Given a polynomial in some implicit representation, it asks whether the polynomial is identically zero or not. It is a fundamental problem in algorithms and complexity theory. It has found applications in algorithm design, for example in algorithms for perfect matching in graphs [17, 36, 39], for primality testing [2, 3, 4], for equivalence testing of read once branching programs [13], and for multi-set equality testing [14], and also in complexity theory, for example, in establishing some major results related to interactive proofs and probabilistically-checkable proofs [38, 9, 8, 7, 45]. In fact, it has also been discovered that an algorithm for polynomial identity testing is intimately connected with complexity theoretic lower bounds [31, 1].


© Markus Bläser and Anurag Pandey;
licensed under Creative Commons License CC-BY

In order to formalize the algorithmic problem of polynomial identity testing, it is important to specify the representation in which the polynomial is given. One possibility is that the polynomial is given as a blackbox, which means that the algorithm is restricted to using the given representation of the given polynomial only as an oracle. That is, the algorithm is only allowed to query the values of the polynomial at points of its choice. Apart from that, the algorithm only knows that the given polynomial comes from some particular class of polynomials. The other possibility is that the algorithm is also allowed to look into the representation. In this case, if the polynomial is given as a list of coefficients, the problem becomes trivial. The problem remains interesting in the case when the polynomial is given in some succinct representation, for example, either as a determinant of a given symbolic matrix, as an algebraic branching program, or more generally, as some arithmetic circuit.

It is known that randomness is necessary for a polynomial time blackbox PIT algorithm (see for example [35]). The challenge thus in this case is to find polynomial time algorithms that use optimal amount of randomness. Randomness is not known to be essential in the setting when the polynomial is given as an arithmetic circuit. In fact, it is popularly believed that there do exist polynomial time algorithms for this version of PIT which do not use randomness. More generally, it is believed that in the regime of efficient computation, randomization is not essential, that is, the complexity classes P and BPP are equal (see [30]). In this case, the challenge is to come up with a deterministic algorithm. A lot of progress has happened over the years towards both the challenges [18, 35, 32, 12, 10, 28, 27, 24, 23, 22, 6, 11, 2, 33, 15, 37, 29], however the problems are still far from the complete solution. For a history on the progress on polynomial identity testing, we refer the readers to [47, 42, 43].

In this work, we are interested in blackbox polynomial identity testing. We will focus our attention to the case when the underlying field is a finite field. More precisely, we are interested in the following computational problem.

▶ **Problem 1.** *Let $(\mathbb{F}_q, n, d)$ denote the class of multivariate polynomials over $\mathbb{F}_q$ in n variables with degree bounded by d* [1] *with $q \geq d+1$. Given a polynomial $p \in (\mathbb{F}_q, n, d)$ as a blackbox and a parameter $\epsilon > 0$, decide whether p is an identically zero polynomial in randomized $poly(n, d)$ time with success probability $1 - \epsilon$.*

We are interested in algorithms for Problem 1 which minimize the number of random bits needed to solve it. In the next subsection we discuss some previous works on the problem that are relevant to this article. While mentioning these works, we will assume the error bound ϵ to be some inverse polynomial in (nd), and we will focus only on algorithms that run in $poly(n, d)$ time under this assumption.

1.1 Previous works on Problem 1

A lot of randomized algorithms are known for PIT in the blackbox setting. The first one is the algorithm due to Schwartz-Zippel-DeMillo-Lipton [2] [44, 49, 21]. It uses $\sum_{i=1}^{n} \log(d_i + 1) + n \log n + 1$ random bits. Then came the algorithm by Lewin and Vadhan [35] which used $\sum_{i=1}^{n} \lceil \log d_i \rceil$ random bits, where d_i refers to the degree of the given polynomial with respect to the variable x_i. Using the Kronecker substitution, Agrawal and Biswas [2] gave a test with $\lceil \sum_{i=1}^{n} \log(d_i + 1) \rceil$ random bits, while Bläser-Hardt-Steurer [12] extended their Kronecker substitution based test to work for asymptotically smaller fields by using $\sum_{i=1}^{n} \log(d_i + 1) + \tilde{O}(\sqrt{\sum_{i=1}^{n} \log(d_i + 1)})$ random bits.

[1] in this paper, unless stated otherwise, degree always refers to the total degree

[2] In their paper [21], DeMillo-Lipton work with the total degree. However, implicitly, the analysis of their algorithm only assumes the individual degrees to be bounded by d.

These works achieve optimal number of random bits in the regime where individual degrees of $x_1, \ldots, x_n$ are bounded by $d_1, \ldots, d_n$ respectively. In that regime, a simple dimension argument shows a lower bound of $\log(\prod_{i=1}^{n}(d_i+1)) - \log T(n, d_1, \ldots, d_n)$, where $T(n, d_1, \ldots, d_n)$ denotes the number of queries made to the blackbox [35, Theorem 7.1]. Thus, when $T(n, d_1, \ldots, d_n)$ is $poly(n)$ bounded, we get a $(1 - o(1)) \sum_{i=1}^{n} \log(d_i + 1)$ lower bound on the number of random bits needed. However, when we are in the setting as given in Problem 1, that is, when only a bound on the total degree is given, the number of random bits used by these methods are asymptotically similar to that of Schwartz-Zippel-DeMillo-Lipton i.e. $\Omega(n \log d)$, which is far from optimal in the regime where $d \ll n$. In this regime, again using a simple dimension argument (see [35, Theorem 7.1] and Lemma 9), we have the following lower bound:

▶ **Fact 1.** *Any blackbox identity testing algorithm against $(\mathbb{F}_q, n, d)$, $q \geq d+1$ which makes $T(n,d)$ queries to the blackbox and succeeds with probability $1-\epsilon$ uses at least $\log(\binom{n+d}{d}) + \log(1/\epsilon) - \log T(n,d)$ random bits.*

Applying Stirling's approximation on $\binom{n+d}{d}$ in the above when $d = o(n)$ gives $\log(\binom{n+d}{d}) = (1+o(1))d\log(\frac{n+d}{d}) = d\log n + o(d\log n)$ [20]. Plugging this in above, with $T(n,d) = (nd)^{O(1)}$, we get the lower bound of $d\log n + \log(1/\epsilon) + o(d\log n)$.[3]

Moving on to the previous works when $d \ll n$, several algorithms are known that actually do achieve the $O(d \log n)$ random bits. For instance, Klivans-Spielman [32], Bogdanov [15], Shpilka-Volkovich generator [46], Lu [37], Guruswami-Xing [29] and finally Bshouty [16] (also see Cohen-Ta-Shma [19]). However, except for [29] and [16], all of them require the field size to be superlinear in d/ϵ as a pre-condition for the algorithm. Moreover, in all of these algorithms including the ones in Bshouty [16] and Guruswami-Xing [29], the number of random bits used is $\geq 2d \log n$.

1.2 Our contributions and methods

From the above, we can see that in the low-degree regime, the number of random bits needed by all the previously discovered algorithms, is away from the information theoretically optimal bound at least by a constant multiplicative factor. We take up the challenge and solve it. We give an algorithm that matches the information theoretic lower bound differing from it only in the lower order terms.

More precisely, we show the following:

▶ **Theorem 1.** *Given a polynomial $f \in (\mathbb{F}_q, n, d)$ with $q \geq d+1$ as a blackbox, and a parameter $\epsilon > 0$, there exists a randomized $poly(n,d)$ time algorithm which uses $d\log n + \log(1/\epsilon) + O(d \log\log n)$ random bits and outputs whether f is an identically zero polynomial with success probability $1-\epsilon$.*

Starting point of our algorithm is an algorithm given in Bshouty [16]. He used the so-called Sidon sets (discussed in Section 2.1) for polynomial identity testing by using them to reduce the problem to the univariate setting while preserving the nonzeroness. He then used the obvious randomized algorithm for the obtained univariate polynomial. This, however, requires the field-size to be large. He gets around this problem by inventing the concepts of testers (discussed in Section 2.2). Informally, testers take a point α from a field $\mathbb{F}$ and map it to a bunch of points in a smaller subfield of $\mathbb{F}$, while maintaining the property that if $f(\alpha) \neq 0$, then f will evaluate to a nonzero value on at least one of the points given by the tester.

[3] this is what we refer to as the information theoretic lower bound in this article

He used two constructions for Sidon sets for this purpose. One of them is not known to be poly time constructible, while the other, which is poly time constructible is factor 2 away from the information theoretic lower bound. To overcome this, we use a new, elementary construction of Sidon sets that is mentioned in Timothy Gowers' weblog [26] (presented in Section 2.1).

Our second contribution is aesthetic in nature. We first remind the readers that a hitting set against a class $\mathcal{P} \subseteq \mathbb{F}[x_1, \ldots, x_n]$ is a set of point $\mathcal{H} \subseteq \mathbb{F}^n$ such that no nonzero polynomial in $\mathcal{P}$ evaluates to zero on all the points in $\mathcal{H}$. We present two simple constructions of information theoretically optimal hitting sets (i.e. of size $\binom{n+d}{d}$) against $(\mathbb{F}, n, d)$ with $|\mathbb{F}| \geq d+1$ that are, at least implicitly, present in the literature. We extract them out and give very simple and neat proofs for both. The first construction (presented in Section 3.1) is essentially the set of exponent vectors of all the monomials spanning $(\mathbb{F}, n, d)$. This works when $\{0, 1, \ldots, d\} \subseteq \mathbb{F}$. The second construction (presented in Section 3.2) says that taking all the intersection points of n-sized subsets of a set of $n+d$ hyperplanes in general position also forms a hitting set against $(\mathbb{F}, n, d)$ of optimal size.

In the rest of the paper, $(\mathbb{F}, n, d)$ (resp. $(\mathbb{F}_q, n, d)$) denotes the class of n-variate polynomials with degree bounded by d over $\mathbb{F}$ (resp. $\mathbb{F}_q$). For a natural number $d \in \mathbb{N}$, $[d]$ denotes the set $\{1, \ldots, d\}$.

2 Polynomial Identity Testing with optimal randomness

In this section, we present our main result. We first describe the main component of the proof, that is, the construction of Sidon sets in Section 2.1, and then describe the way to reduce the field size in Section 2.2. We finally give our algorithm and the proof for our main theorem in Section 2.3.

2.1 Sidon Sets

A set $\mathcal{S} := \{s_1, s_2, \ldots, s_n\} \subset \mathbb{Z}_{\geq 0}$ is said to be a Sidon B_d set if every element in the set $d\mathcal{S} := \{s_{i_1} + s_{i_2} + \ldots + s_{i_d} \mid \forall k \in [d], s_{i_k} \in \mathcal{S}\}$ are distinct up to rearrangements of the summands. We also have a stronger notion: we call $\mathcal{S}$ to be Sidon $B_{\leq d}$ set if the sums $\{s_{i_1} + s_{i_2} + \ldots + s_{i_r},\ r \leq d \mid \forall k \in [r], s_{i_k} \in \mathcal{S}\}$ are distinct up to rearrangements of the summands. For our purposes, the stronger notion of Sidon $B_{\leq d}$ set when $d \ll n$ will be useful. We are interested in constructions that minimize the size of the maximum element of $\mathcal{S}$ and are $poly(n, d)$ time constructible.

Sidon sets and its variants have a long history in mathematics and several explicit constructions are known. We refer the readers to a survey by Kevin O'Bryant [40].

In complexity theory, explicit Sidon set constructions have also been used, for example, by Bshouty for constructions of hitting sets for black box polynomial identity testing [16], and by Kumar and Volk for matrix factorization lower bounds [34].

Bshouty uses two constructions for polynomial identity testing. The first construction uses discrete log and is not known to be poly time constructible [16, Lemma 59]. The second construction is poly time constructible, but the value of the maximum element is $(2nd)^{2d}$ [16, Lemma 60]. This $2d$ in the exponent makes this construction suboptimal for our purposes because the resulting randomized PIT algorithm will have $\geq 2d \log n$ random bits, which is factor 2 away from the information theoretic bound in low degree regime which is the regime of interest in this paper.

This motivated us to look for constructions that are both polynomial time constructible and also give rises to PIT algorithm with optimal randomness. That is when we stumbled across the weblog of Timothy Gowers about the so-called dense Sidon sets [26] where he describes the idea of a construction by Imre Z. Ruzsa [41] that scales up for our purposes too.

In its core, the construction is based on the fundamental theorem of arithmetic. Informally, when we take a set of primes and consider two different multi-subsets of them. Then the product of elements will be different for the two multi-subsets. Now taking logarithm of products convert them to sums. These simple facts along with the mean value theorem constitute the ingredients of the proof of the construction. We give the construction now.

▶ **Theorem 2.** *For every n, d, there exists a poly(n, d) time constructible Sidon $B_{\leq d}$ set $S_{n,d} := \{b_1, \ldots, b_n\} \subset \mathbb{N}$ with $b_1 < b_2 < \cdots < b_n$, with $b_n \leq \lceil (d+1) \cdot (2n \log n)^d \cdot \log(2n \log n) \rceil$.*

Proof. We take the first n primes $p_1, \ldots, p_n$. By prime number theorem, we know that $p_n < n(\log n + \log \log n) < 2n \log n$. Let $I, J \subseteq \{1, \ldots, n\}$ be multisets, where $|I|, |J| \leq d$, $I \neq J$. By the fundamental theorem of arithmetic, we have $\prod_{i \in I} p_i \neq \prod_{j \in J} p_j$. Without loss of generality, we can assume that $\prod_{i \in I} p_i < \prod_{j \in J} p_j$, that is, $\prod_{i \in I} p_i \leq \prod_{j \in J} p_j + 1$. Now applying the mean value theorem on the function $f(x) = \log x$ in the interval $[a, b]$ with $a := \prod_{i \in I} p_i$ and $b := \prod_{j \in J} p_j$, we get that

$$\sum_{j \in J} \ell_j - \sum_{i \in I} \ell_i = \frac{1}{c} \left(\prod_{j \in J} p_j - \prod_{i \in I} p_i \right), \text{ for some } c \in (a, b), \text{ where } \ell_k := \log p_k.$$

The numerator in the RHS of the equation is at least 1, while the denominator is upper bounded by $b = \prod_{j \in J} p_j$. Thus, we have

$$\sum_{j \in J} \ell_j - \sum_{i \in I} \ell_i \geq \frac{1}{\prod_{j \in J} p_j}. \tag{1}$$

Thus, if we choose the set to be set of logarithm of the first n primes, we do get, that for distinct multi-subsets of size at most d, the sum of elements are also distinct. However, clearly, the elements and their differences will not be all integers. But the above calculation is suggestive of what the set should be. Note that in Equation (1), the denominator of the RHS, that is, $\prod_{j \in J} p_j$ is upper bounded by $(2n \log n)^d$, Thus,

$$\sum_{j \in J} (d+1) \cdot \ell_j \cdot (2n \log n)^d - \sum_{i \in I} (d+1) \cdot \ell_i \cdot (2n \log n)^d \geq d + 1$$

Now, if we consider the set $\mathcal{S}_{n,d}$ of size n with elements being positive integers $b_k := \lceil (d+1) \cdot \ell_k \cdot (2n \log n)^d \rceil$ of size n, we have that $\sum_{j \in J} b_j - \sum_{i \in I} b_i > 0$. Thus, $\mathcal{S}_{n,d}$ is a Sidon $B_{\leq d}$ set.

It only remains to argue that the construction can be done in $poly(n, d)$ time. We need to show that all the $b_k = \lceil (d+1) \cdot \log p_k \cdot (2n \log n)^d \rceil$ are $poly(n, d)$ time constructible, It is known that the first n primes are easily constructible, for example, by using Sieve of Eratosthenes which takes $O(n \log \log n)$ time. The other functions like log and powering function are also known to be efficiently computable to the desired precision. ◀

We now present the concept useful for transferring a polynomial identity testing algorithm over a large field to an algorithm for a small subfield of it.

2.2 Testers

The notion of testers is also crucial for our algorithm. They were introduced by Bshouty in [16]. He also used it for several applications including in the setting of blackbox PIT. We will be using it in the same fashion as he did i.e. to reduce the field size of the blackbox PIT set that we would be using for the algorithm. We present the definition of testers restricted to the setting that we need. He defined it for a more general setting.

▶ **Definition 3.** *Let $\mathbb{F}_q$ be a finite field with q elements and let $\mathbb{F}_{q^{t_1}}$ and $\mathbb{F}_{q^{t_2}}$ be two field extensions of $\mathbb{F}_q$ viewed as $\mathbb{F}_q$-algebras with $t_1 \geq t_2$, and let $\mathcal{P} \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ be a class of multivariate polynomials. Let $L = \{\ell_1, \ldots, \ell_\nu\}$ be a set of maps $\mathbb{F}_{q^{t_1}}^n \to \mathbb{F}_{q^{t_2}}^n$. For $f \in \mathcal{P}$, we denote by fL the map $\mathbb{F}_{q^{t_1}}^n \to \mathbb{F}_{q^{t_2}}^\nu$ defined as: for $\boldsymbol{a} \in \mathbb{F}_{q^{t_1}}^n$, $(fL)(\boldsymbol{a}) = (f(\ell_1(\boldsymbol{a})), \ldots, f(\ell_\nu(\boldsymbol{a})))$. We say that L is an $(\mathcal{P}, \mathbb{F}_{q^{t_1}}, \mathbb{F}_{q^{t_2}})$-tester if for every $\boldsymbol{a} \in \mathbb{F}_{q^{t_1}}^n$ and $f \in \mathcal{P}$ we have*

$$(fL)(\boldsymbol{a}) = \boldsymbol{0} \implies f(\boldsymbol{a}) = 0.$$

The size of the tester L is defined as $|L| = \nu$, the number of maps constituting L.

So, essentially, a tester L for the class of polynomials $\mathcal{P}$ is a set of maps from a field to its subfield such that for every point on which a polynomial $f \in \mathcal{P}$ evaluates to a nonzero value, the tester gives a set of points in the subfield such that the polynomial evaluates to a nonzero value on at least one of the points given by the tester.

Hence a tester is very useful for reducing a blacbox PIT set over a bigger field to a blackbox PIT set over a smaller field while incurring a blowup by the size of the tester. Bshouty [16] also gave many constructions of testers against several classes of multivariate polynomials which helped him achieve constructions of hitting sets which are optimal with respect to the field size and the size of hitting sets.

The tester that is relevant to our purposes which we will be using as a blackbox has the following property. For a proof we refer the readers to [16].

▶ **Lemma 4** ([16], Theorem 40)**.** *Let $\mathcal{P} := (\mathbb{F}_q, n, d) \subseteq \mathbb{F}_q[x_1, \ldots, x_n]$ denote the class of n-variate, degree d polynomials over $\mathbb{F}_q$, with $q \geq d + 1$. Then, for every n, d, t, there exists a $(\mathcal{P}, \mathbb{F}_{q^t}, \mathbb{F}_q)$-tester L of size $O(d^5 t)$ that can be constructed in $poly(n, d, t)$ time.*

The above lemma clearly suggests a strategy for construction of blackbox PIT sets: first design a blackbox PIT set over a large extension field and then reduce the field size to $d + 1$ using the tester promised by the above lemma.

2.3 The algorithm: Proof of Theorem 1

In this section, we present our randomized algorithm for polynomial identity testing and prove Theorem 1.

Before we prove the theorem, we state a simple lemma about univariate polynomials that we will need in the proof.

▶ **Lemma 5.** *Let $f \in \mathbb{F}_q[x]$ be a nonzero univariate polynomial whose degree is bounded by d. Let $\mathbb{F}_{q^t}$ be an extension field of $\mathbb{F}_q$ such that $|\mathbb{F}_{q^t}| \geq d/\epsilon$ and $\boldsymbol{a}$ is sampled uniformly at random from $\mathbb{F}_{q^t}$, then $f(\boldsymbol{a}) \neq 0$ with probability $1 - \epsilon$.*

Lemma 5 follows from the folklore theorem that a univariate polynomial of degree d over a field $\mathbb{F}_q$ has at most d roots in any field extension $\mathbb{F}_{q^t}$ of $\mathbb{F}_q$.

We are now ready to prove Theorem 1.

▶ **Theorem 6** (Theorem 1 restated)**.** *Given a polynomial $f \in (\mathbb{F}_q, n, d)$ with $q \geq d + 1$ as a blackbox or as a $poly(n, d)$-sized arithmetic circuit, and $\epsilon > 0$, there exists a randomized $poly(n, d)$ time algorithm which uses $d \log n + \log(1/\epsilon) + O(d \log \log n)$ random bits and succeeds with probability $1 - \epsilon$.*

Proof. Suppose we are given a polynomial $f \in (\mathbb{F}_q, n, d)$ as a blackbox. To test whether the given polynomial is an identically zero polynomial or not, our algorithm works as follows:

Step 1. Construct Sidon set: Given n, d, we construct a Sidon $B_{\leq d}$ set $\mathcal{S}_{n,d} = \{b_1, \ldots, b_n\}$ using the construction in Theorem 2.

Step 2. Pick a random point from large field: We pick a random point α from the field $\mathbb{F}_{q^t}$ with $t = \lceil \log_q((b_n d)/\epsilon) \rceil$.

Step 3. Construct the tester: Next we construct a $(\mathcal{P}, \mathbb{F}_{q^t}, \mathbb{F}_q)$-tester, $L = \{\ell_1, \ldots, \ell_\nu\}$, for $\mathcal{P} = (\mathbb{F}_q, n, d)$ and $t = \lceil \log_q((b_n d)/\epsilon) \rceil$ using the construction promised by Lemma 4.

Step 4. Reduce the field size by testers: We then apply the tester L on $(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_{q^t}^n$ to get the set of points $\ell_1(\alpha^{b_1}, \ldots, \alpha^{b_n}), \ldots, \ell_\nu(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_q^n$.

Step 5. Evaluate: We evaluate f on $\ell_1(\alpha^{b_1}, \ldots, \alpha^{b_n}), \ldots, \ell_\nu(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_q^n$. If f evaluates to zero on $\ell_k(\alpha^{b_1}, \ldots, \alpha^{b_n})$, for every $k \in 1, \ldots, \nu$, we output that f is an identically zero polynomial. Otherwise we output that f is not a zero polynomial.

We now show the correctness of the above algorithm. The Sidon $B_{\leq d}$ set $\mathcal{S}_{n,d} = \{b_1, \ldots, b_n\}$, $b_1 < b_2 < \cdots < b_n$ and $b_n = \lceil (d+1) \cdot (2n \log n)^d \cdot \log(2n \log n) \rceil$ from Theorem 2 is used to reduce the problem to the univariate case. It is also $poly(n, d)$ time constructible. By the definition of Sidon $B_{\leq d}$ set in Section 2.1, it follows that for distinct multi-subsets of $\mathcal{S}_{n,d}$, the sum of elements will also be distinct. Thus, the map $(x_1, x_2, \ldots, x_n) \mapsto (x^{b_1}, x^{b_2}, \ldots, x^{b_n})$ maps the monomials of the degree at most d in the variables $x_1, \ldots, x_n$ to distinct univariate monomials in x. In particular, every nonzero polynomial $f \in (\mathbb{F}_q, n, d)$ maps to a nonzero polynomial $g \in (\mathbb{F}_q, 1, b_n d)$. Thus, g is a polynomial of degree bounded by $b_n d$.

Now, by Lemma 5, on a randomly chosen point α from the extension field $\mathbb{F}_{q^t}$ with $|\mathbb{F}_{q^t}| \geq (b_n d)/\epsilon$, g will evaluate to a nonzero value with probability $\geq 1 - \epsilon$. Hence, f will evaluate to a non-zero value on $(\alpha^{b_1}, \ldots, \alpha^{b_n})$ with probability $\geq 1 - \epsilon$. The number of random bits needed is $\log((b_n d)/\epsilon) = \log b_n + \log d + \log(1/\epsilon) = d \log n + O(d \log \log n) + \log(1/\epsilon)$ as claimed.

Finally we use an $((\mathbb{F}_q, n, d), \mathbb{F}_{q^t}, \mathbb{F}_q)$-tester from Lemma 4 on $(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_{q^t}^n$ to get the set of points $\ell_1(\alpha^{b_1}, \ldots, \alpha^{b_n}), \ldots, \ell_\nu(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_q^n$. By Lemma 4, the number of points $\nu = O(d^5 t) = O(d^5(\log(b_n d) + \log(1/\epsilon))) = O(d^6(\log n + \log 1/\epsilon))$ and time to construct the tester is $poly(n, d, t) = poly(n, d, \log 1/\epsilon)$. By the definition of testers from Definition 3, for a nonzero polynomial $f \in (\mathbb{F}_q, n, d)$, if $f(\alpha^{b_1}, \ldots, \alpha^{b_n}) \neq 0$, then at least one of $f(\ell_1(\alpha^{b_1}, \ldots, \alpha^{b_n})), \ldots, f(\ell_\nu(\alpha^{b_1}, \ldots, \alpha^{b_n}))$ also evaluates to a nonzero value. Thus, we get the desired result. ◀

▶ Remark 7. When $d = o(\log n)$, we can get an improvement on the number of random bits from $d \log n + d \log \log n +$ lower order terms (as in Theorem 1) to $d \log n + d \log d +$ lower order terms. This can be achieved by adapting an idea due to Goldwasser-Grossman [25, Lemma 8] that they used to construct weight assignments to get unique min-weight perfect matching. Their idea suggests a map $(x_1, x_2, \ldots, x_n) \mapsto (x^{b_1}, x^{b_2}, \ldots, x^{b_n})$, where $b_i = [i]_p + (pd)[i^2]_p + (pd)^2[i^3]_p + \cdots + (pd)^d[i^{d+1}]_p$, where $[x]_p$ denotes the number between 0 and $p-1$ which is equal to x modulo p, and p is a prime number greater than n. This map can replace the map due to Sidon Sets in the Step 1 from the above proof, while the rest of the algorithm and the proof remains the same. As in the above proof of Theorem 1, the number of random bits needed equals $\log b_n + \log d + \log(\frac{1}{\epsilon})$ which becomes $d \log n + d \log d +$ lower order terms. For the correctness of this map, we refer the reader to the proof of Lemma 8 in [25].

▶ Remark 8. Our algorithm works for fields of zero characteristic as well. In fact, whenever the field size is already larger than $(b_n d)/\epsilon$, we do not need Step 3 and Step 4, and we can directly evaluate f on $(\alpha^{b_1}, \ldots, \alpha^{b_n})$ in Step 5. Thus, we have an algorithm for blackbox polynomial identity testing which uses information theoretically optimal number of random bits for all fields $\mathbb{F}$ with $|\mathbb{F}| \geq d + 1$.

3 Optimal Hitting sets

In this section, we describe a few optimal hitting sets, i.e. the ones that exactly matches with the lower bound against the class of n-variate polynomials with total degree bounded by d. The authors are excited about these proofs because of their simplicity and elegance.

We first begin by stating the straight-forward folklore lower bound.

▶ **Lemma 9.** *For any hitting set $\mathcal{H}$ against the class n-variate polynomials with total degree bounded by d over a field $\mathbb{F}$, we have $|\mathcal{H}| \geq \binom{n+d}{d}$.*

This follows by the fact that the set of all n-variate polynomials with total degree bounded by d over a field $\mathbb{F}$ forms a vector space of dimension $\binom{n+d}{d}$. This is true because the number of monomials supported on n-variables with total degree bounded by d is $\binom{n+d}{d}$, and they form the basis for the vector space as they are all $\mathbb{F}$-linearly independent, and all polynomials in the set can be represented as an $\mathbb{F}$-linear combination of them. Thus, in the worst case, one needs to query f on at least $\binom{n+d}{d}$ points.

We now consider a very popular construction which is suboptimal for our purposes.

Construction 0 – Schwartz-Zippel-DeMillo-Lipton lemma: As a consequence of the Lemma [44, 49, 21], for $(\mathbb{F}, n, d)$, one gets the hitting set $\mathcal{H}_0 := S^n$ where $S \subseteq \mathbb{F}$, with $|S| = d + 1$, which is the grid of size $(d+1)^n$. We point out that this is optimal when we are considering the set of n-variate polynomials with *individual degrees* of each variable bounded by d, by a similar argument as in Lemma 9. However, this is not optimal for $(\mathbb{F}, n, d)$ where we bound the *total degree*. Especially when $d \ll n$, the gap is huge.

Thus, it makes sense to investigate optimal hitting sets for $(\mathbb{F}, n, d)$. In what follows, we present two such constructions for the hitting set. They were both, at least implicitly, already present in the literature. We also believe that other constructions can be found without much effort. However our predilection towards these constructions and our new proofs is purely aesthetic.

3.1 Construction 1: The set of exponent vectors

The lower bound tells that any hitting set $\mathcal{H}$ should have size at least $\binom{n+d}{d}$. Now this $\binom{n+d}{d}$ comes from the number of monomials in n variables of degree at most d. Very interestingly, when $\{0, 1, \ldots, d\} \subseteq \mathbb{F}$, these monomials also suggest a set of points of size $\binom{n+d}{d}$ that can be seen as a potential hitting set: simply collect all the exponent vectors of all the monomials in a set, viewing them as points in $\mathbb{F}^n$, that is, the suggested set is $\mathcal{H}_1 := \{(x_1, \ldots, x_n) \in \{0, 1, \ldots, d\}^n \mid x_1 + x_2 + \cdots + x_n \leq d\}$. The above construction obviously requires that $\mathbb{F}^n$ indeed contains $\mathcal{H}_1$ as a subset. Surprisingly to some, and beautifully to the authors, this indeed works. This was shown by Bshouty [16, Lemma 77] using Combinatorial Nullstellensatz [5].

In this work, we give a direct inductive proof which we found with Mrinal Kumar. It bypasses the combinatorial nullstellensatz and flows along the lines of the proof of Schwartz-Zippel-DeMillo-Lipton lemma. We are suprised that we did not find this proof somewhere in the literature.

▶ **Theorem 10.** *If $\{0, 1, \ldots, d\} \subseteq \mathbb{F}$, then the set $\mathcal{H}_1 := \{(x_1, \ldots, x_n) \in \{0, 1, \ldots, d\}^n \mid x_1 + x_2 + \cdots + x_n \leq d\}$, is a hitting set for $(\mathbb{F}, n, d)$.*

Proof. Consider the integral grid $\mathcal{G} := \{0, 1, \dots d\}^n$ with $|\mathcal{G}| = (d+1)^n$. Now the statement of the theorem can be rephrased as: for every nonzero polynomial $f \in (\mathbb{F}, n, d)$, there exists a point $g \in \mathcal{G}$ with its ℓ_1-norm $\|g\|_1 \leq d$, such that $f(g) \neq 0$. We use this as our induction hypothesis and prove it by induction on the number of variables n.

For $n = 1$, that is the univariate case, this holds true because every degree d polynomial has at most d zeros. Suppose the hypothesis holds for $n-1$. For the inductive step, write a nonzero $f \in (\mathbb{F}, n, d)$ as a univariate in x_n as $f = \sum_{i=1}^{d'} P_i(x_1, \dots, x_{n-1}) x_n^i$, where d' is the maximum degree of f in x_n and $P_i(x_1, \dots, x_{n-1})$ are the coefficients coming from $\mathbb{F}[x_1, \dots, x_{n-1}]$. Now consider $P_{d'}(x_1, \dots, x_{n-1})$ which is the coefficient of the highest degree term in x_n. If f is a nonzero polynomial, then so is $P_{d'}(x_1, \dots, x_{n-1})$. Also, $\deg(P_{d'})$ is bounded by $d - d'$. By the induction hypothesis, there is a point s' in the grid $\mathcal{G}' := \{0, 1, \dots, d - d'\}^{n-1}$ with $\|s'\|_1 \leq d - d'$ such that $P_{d'}(s') \neq 0$. Now we fix $x_1, \dots, x_{n-1}$ to the values as given by s'. Now $P_{d'}$ restricted to the assignment s' is a univariate polynomial in x_n of degree d'. Thus, setting x_n to a value in $\{0, 1, \dots d'\}$ gives a point on which f evaluates to nonzero. The ℓ_1 norm of the point is at most $(d - d') + d' = d$. ◀

We now give another construction which we find beautiful.

3.2 Construction 2: Intersection of hyperplanes in general position

The construction is as follows: In the projective space $\mathbb{P}^n(\mathbb{F})$ over a field $\mathbb{F}$, with $|\mathbb{F}| \geq d + 1$, take a collection $\mathcal{C}$ of $n + d$ hyperplanes in general position i.e. every size n subsets of $\mathcal{C}$ intersect in a point, whereas no size $n + 1$ subset of $\mathcal{C}$ intersect. Now all intersection points of n-sized subset of $\mathcal{C}$ gives the desired hitting set.

We now mention a standard explicit family of hyperplanes in general position.

▶ **Example 11.** Take hyperplanes $H_1, \dots, H_{n+d}$ in the projective space $\mathbb{P}^n$ where H_i is given by the equation $t_i^1 x_1 + t_i^2 x_2 + \cdots + t_i^n x_n + x_{n+1} = 0$, where $t_i \in \mathbb{F}$. Then, $H_1, \dots, H_{n+d}$ are hyperplanes in general position.

Itaï Ben Yaacov [48] considers hyperplanes in general position and gives an algebraic proof of a generalized Vandermonde Identity in higher dimension. What his identity implies is that taking all intersection points of n-sized subsets of $n+d$ hyperplanes in general position gives a hitting set for $(\mathbb{F}, n, d)$.

In order to state his result, we need some notations. Let $M_{p\times q}(\mathbb{F})$ denote the set of all $p \times q$ matrices over $\mathbb{F}$. He defines the following three maps. It useful to think that the $(n+1) \times m$ matrix Q is denoting the family of m hyperplanes, say $\mathcal{H}_m$ in $\mathbb{P}^n$, i.e. the entries of each column correspond to the coefficients of a hyperplane.

- **i)** $\mu : M_{(n+1)\times m}(\mathbb{F}) \to M_{(n+1)\times \binom{m}{n}}(\mathbb{F})$ sends an $(n+1) \times m$ matrix Q to a matrix P whose entries are all the minors of Q of order n. Note that a lexicographic ordering on the chosen sequence of rows and columns of Q induces an ordering on the minors as well. By Cramer's rule from linear algebra, P is precisely the matrix of intersection points of all n-sized subsets of $\mathcal{H}_m$, where each column of Q has the coordinates of an intersection point as its entries.
- **ii)** $\delta : M_{(n+1)\times m}(\mathbb{F}) \to \mathbb{F}$ sends an $(n+1) \times m$ matrix Q to the product of all its minors of order $n + 1$.
- **iii)** $\nu_r : M_{(n+1)\times m}(\mathbb{F}) \to M_{\binom{n+r}{n}\times m}(\mathbb{F})$ applies the Veronese map on each column i.e. for each column, it applies all the n-variate degree r monomials on the entries of the column. Assume an ordering (say, inverse lexicographical ordering) on the monomials.

We are now ready to quote the generalized Vandermonde identity in higher dimension.

▶ **Theorem 12** ([48], Theorem 1.4)**.** *Let R be a commutative ring. $n \leq m \in \mathbb{N}$, and let Q be a $(n+1) \times m$ matrix over R. Then $\nu_{m-n}\mu Q$ is a square matrix of order $\binom{m}{n}$, and the following Vandermonde identity of order m in dimension n holds:*

$$\det(\nu_{m-n}\mu Q) = (\delta Q)^n$$

Then the above theorem with $m = n + d$, implies that if $(\delta Q) \neq 0$, which is the algebraic condition for the $n + d$ hyperplanes to be in general position, then $\det(\nu_{n+d-n}\mu Q) \neq 0$. Now $\nu_{m-n}\mu Q$ is the matrix with Veronese map applied on the intersection points of n-sized subsets of m hyperplanes. Normalizing the coordinates by the last coordinate gives us the points in the affine setting with the Veronese maps essentially applying all the monomials of degree at most d on the points. Thus, $\det(\nu_d \mu Q) \neq 0$ means that the set of intersection points form a hitting set against $(\mathbb{F}, n, d)$.

We now present a direct, simple, geometric proof of the construction which we found with Raimund Seidel.

▶ **Theorem 13.** *Let $H_1, \ldots, H_{n+d}$ be hyperplanes in general position. If $f \in (\mathbb{F}, n, d)$ vanishes on all the points obtained by intersecting all n-sized subsets of $\{H_1, \ldots, H_{n+d}\}$, then f is an identically zero polynomial.*

Proof. We prove the above statement by induction on the number of variables n. The base case $n = 1$ is the univariate case and the hyperplanes become $d+1$ points, and the statement of the lemma reduces to f vanishing on $d+1$ points. Thus, the statement holds in this case because a degree d univariate polynomial that vanishes on $d+1$ points is an identically zero polynomial.

Suppose that the statement holds for the number of variables up to $n-1$, and we assume an $f \in (\mathbb{F}, n, d)$ that vanishes on all the intersection points of n-sized subsets of $\{H_1, \ldots, H_{n+d}\}$. The assumption, in particular, implies that f restricted to the hyperplane H_1 vanishes on all the intersection points of $(n-1)$-sized subsets of $\{H_2, \ldots, H_{n+d}\}$. However, note that f restricted to H_1 reduces to an $(n-1)$-variate case and hence we can apply the induction hypothesis and conclude that f restricted to H_1 is identically zero. Doing the same for all the hyperplanes, we get that f restricted to all the hyperplanes $H_1, \ldots, H_{n+d}$ is identically zero. It remains to conclude that f is indeed identically zero. For this, restrict f to a generic line ℓ. Note that $H_1, \ldots, H_{n+d}$ all intersect ℓ at distinct points. Thus, f restricted to ℓ is a univariate which vanishes on $n + d$ points, hence f restricted to a generic ℓ is identically zero. Hence f is identically zero. ◀

Note that an explicit construction corresponding to Theorem 13 can be obtained using the family given in Example 11.

References

1 Manindra Agrawal. Proving lower bounds via pseudo-random generators. In *FSTTCS 2005: Foundations of software technology and theoretical computer science*, volume 3821 of *Lecture Notes in Comput. Sci.*, pages 92–105. Springer, Berlin, 2005. doi:10.1007/11590156_6.

2 Manindra Agrawal and Somenath Biswas. Primality and identity testing via Chinese remaindering. *J. ACM*, 50(4):429–443, 2003. doi:10.1145/792538.792540.

3 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004. doi:10.4007/annals.2004.160.781.

4 Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. Errata: PRIMES is in P. *Ann. of Math. (2)*, 189(1):317–318, 2019. doi:10.4007/annals.2019.189.1.6.

5 NOGA ALON. Combinatorial nullstellensatz. *Combinatorics, Probability and Computing*, 8(1-2):7–29, 1999. doi:10.1017/S0963548398003411.

6 Matthew Anderson, Michael A. Forbes, Ramprasad Saptharishi, Amir Shpilka, and Ben Lee Volk. Identity testing and lower bounds for read-k oblivious algebraic branching programs. In *31st Conference on Computational Complexity*, volume 50 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 30, 25. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2016.

7 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, 1998. doi:10.1145/278298.278306.

8 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of NP. *J. ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.

9 László Babai, Lance Fortnow, and Carsten Lund. Nondeterministic exponential time has two-prover interactive protocols. In *31st Annual Symposium on Foundations of Computer Science, Vol. I, II (St. Louis, MO, 1990)*, pages 16–25. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990. doi:10.1109/FSCS.1990.89520.

10 Markus Bläser and Christian Engels. Randomness efficient testing of sparse black box identities of unbounded degree over the reals. In *28th International Symposium on Theoretical Aspects of Computer Science*, volume 9 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 555–566. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2011.

11 Markus Bläser, Moritz Hardt, Richard J. Lipton, and Nisheeth K. Vishnoi. Deterministically testing sparse polynomial identities of unbounded degree. *Inform. Process. Lett.*, 109(3):187–192, 2009. doi:10.1016/j.ipl.2008.09.029.

12 Markus Bläser, Moritz Hardt, and David Steurer. Asymptotically optimal hitting sets against polynomials. In *Automata, languages and programming. Part I*, volume 5125 of *Lecture Notes in Comput. Sci.*, pages 345–356. Springer, Berlin, 2008. doi:10.1007/978-3-540-70575-8_29.

13 Manuel Blum, Ashok K. Chandra, and Mark N. Wegman. Equivalence of free Boolean graphs can be decided probabilistically in polynomial time. *Inform. Process. Lett.*, 10(2):80–82, 1980. doi:10.1016/S0020-0190(80)90078-2.

14 Manuel Blum and Sampath Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, January 1995. doi:10.1145/200836.200880.

15 Andrej Bogdanov. Pseudorandom generators for low degree polynomials. In *STOC'05: Proceedings of the 37th Annual ACM Symposium on Theory of Computing*, pages 21–30. ACM, New York, 2005. doi:10.1145/1060590.1060594.

16 Nader H. Bshouty. Testers and their applications. *Electronic Colloquium on Computational Complexity (ECCC)*, 19:11, 2012. URL: http://eccc.hpi-web.de/report/2012/011.

17 Suresh Chari, Pankaj Rohatgi, and Aravind Srinivasan. Randomness-optimal unique element isolation with applications to perfect matching and related problems. *SIAM J. Comput.*, 24(5):1036–1050, 1995. doi:10.1137/S0097539793250330.

18 Zhi-Zhong Chen and Ming-Yang Kao. Reducing randomness via irrational numbers. In *STOC '97 (El Paso, TX)*, pages 200–209. ACM, New York, 1999.

19 Gil Cohen and Amnon Ta-Shma. Pseudorandom generators for low degree polynomials from algebraic geometry codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 20:155, 2013. URL: http://eccc.hpi-web.de/report/2013/155.

20 Shagnik Das. A brief note on estimates of binomial coefficients. http://page.mi.fu-berlin.de/shagnik/notes/binomials.pdf. Accessed on 2020-02-18.

21 Richard A. DeMillo and Richard J. Lipton. A probabilistic remark on algebraic program testing. *Inf. Process. Lett.*, 7(4):193–195, 1978. doi:10.1016/0020-0190(78)90067-4.

22 Michael A. Forbes, Sumanta Ghosh, and Nitin Saxena. Towards blackbox identity testing of log-variate circuits. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 54, 16. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

23 Michael A. Forbes and Amir Shpilka. Explicit Noether normalization for simultaneous conjugation via polynomial identity testing. In *Approximation, randomization, and combinatorial optimization*, volume 8096 of *Lecture Notes in Comput. Sci.*, pages 527–542. Springer, Heidelberg, 2013. doi:10.1007/978-3-642-40328-6_37.

24 Michael A. Forbes and Amir Shpilka. Quasipolynomial-time identity testing of noncommutative and read-once oblivious algebraic branching programs. In *2013 IEEE 54th Annual Symposium on Foundations of Computer Science—FOCS 2013*, pages 243–252. IEEE Computer Soc., Los Alamitos, CA, 2013. doi:10.1109/FOCS.2013.34.

25 Shafi Goldwasser and Ofer Grossman. Bipartite perfect matching in pseudo-deterministic NC. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 87, 13. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2017.

26 Timothy Gowers. Gowers weblog: What are dense sidon subsets of {1,2,...,n} like? https://gowers.wordpress.com/2012/07/13/what-are-dense-sidon-subsets-of-12-n-like/. Published on 2012-07-13.

27 Rohit Gurjar, Arpita Korwar, and Nitin Saxena. Identity testing for constant-width, and any-order, read-once oblivious arithmetic branching programs. *Theory Comput.*, 13:Paper No. 2, 21, 2017. doi:10.4086/toc.2017.v013a002.

28 Rohit Gurjar, Arpita Korwar, Nitin Saxena, and Thomas Thierauf. Deterministic identity testing for sum of read-once oblivious arithmetic branching programs. *Comput. Complexity*, 26(4):835–880, 2017. doi:10.1007/s00037-016-0141-z.

29 Venkatesan Guruswami and Chaoping Xing. Hitting sets for low-degree polynomials with optimal density. In *IEEE 29th Conference on Computational Complexity, CCC 2014, Vancouver, BC, Canada, June 11-13, 2014*, pages 161–168. IEEE Computer Society, 2014. doi:10.1109/CCC.2014.24.

30 Russell Impagliazzo and Avi Wigderson. P = BPP if E requires exponential circuits: derandomizing the XOR lemma. In *STOC '97 (El Paso, TX)*, pages 220–229. ACM, New York, 1999.

31 Valentine Kabanets and Russell Impagliazzo. Derandomizing polynomial identity tests means proving circuit lower bounds. In *Proceedings of the Thirty-Fifth Annual ACM Symposium on Theory of Computing*, pages 355–364. ACM, New York, 2003. doi:10.1145/780542.780595.

32 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. ACM, New York, 2001. doi:10.1145/380752.380801.

33 Mrinal Kumar, Ramprasad Saptharishi, and Anamay Tengse. Near-optimal bootstrapping of hitting sets for algebraic circuits. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 639–646. SIAM, Philadelphia, PA, 2019. doi:10.1137/1.9781611975482.40.

34 Mrinal Kumar and Ben Lee Volk. Lower bounds for matrix factorization. *CoRR*, abs/1904.01182, 2019. arXiv:1904.01182.

35 Daniel Lewin and Salil Vadhan. Checking polynomial identities over any field: towards a derandomization? In *STOC '98 (Dallas, TX)*, pages 438–447. ACM, New York, 1999.

36 L. Lovász. On determinants, matchings, and random algorithms. In *Fundamentals of computation theory (Proc. Conf. Algebraic, Arith. and Categorical Methods in Comput. Theory, Berlin/Wendisch-Rietz, 1979)*, volume 2 of *Math. Res.*, pages 565–574. Akademie-Verlag, Berlin, 1979.

37 C. Lu. Hitting set generators for sparse polynomials over any finite fields. In *2012 IEEE 27th Conference on Computational Complexity*, pages 280–286, June 2012. doi:10.1109/CCC.2012.20.

38 Carsten Lund, Lance Fortnow, Howard Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. In *31st Annual Symposium on Foundations of Computer Science, Vol. I, II (St. Louis, MO, 1990)*, pages 2–10. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990. doi:10.1109/FSCS.1990.89518.

39 Ketan Mulmuley, Umesh V. Vazirani, and Vijay V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987. doi:10.1007/BF02579206.

40 Kevin O'Bryant. A complete annotated bibliography of work related to sidon sequences. *The Electronic Journal of Combinatorics [electronic only]*, DS11:39 p., electronic only–39 p., electronic only, 2004. URL: http://eudml.org/doc/129129.

41 Imre Z. Ruzsa. An infinite Sidon sequence. *J. Number Theory*, 68(1):63–71, 1998. doi:10.1006/jnth.1997.2192.

42 Nitin Saxena. Progress on polynomial identity testing. *Bulletin of the EATCS*, 99:49–79, 2009.

43 Nitin Saxena. Progress on polynomial identity testing-II. In *Perspectives in computational complexity*, volume 26 of *Progr. Comput. Sci. Appl. Logic*, pages 131–146. Birkhäuser/Springer, Cham, 2014.

44 J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. Assoc. Comput. Mach.*, 27(4):701–717, 1980. doi:10.1145/322217.322225.

45 Adi Shamir. IP = PSPACE. In *31st Annual Symposium on Foundations of Computer Science, Vol. I, II (St. Louis, MO, 1990)*, pages 11–15. IEEE Comput. Soc. Press, Los Alamitos, CA, 1990. doi:10.1109/FSCS.1990.89519.

46 Amir Shpilka and Ilya Volkovich. Improved polynomial identity testing for read-once formulas. In *Approximation, randomization, and combinatorial optimization*, volume 5687 of *Lecture Notes in Comput. Sci.*, pages 700–713. Springer, Berlin, 2009. doi:10.1007/978-3-642-03685-9_52.

47 Amir Shpilka and Amir Yehudayoff. Arithmetic circuits: a survey of recent results and open questions. *Found. Trends Theor. Comput. Sci.*, 5(3-4):207–388 (2010), 2009. doi:10.1561/0400000039.

48 Itaï Ben Yaacov. The vandermonde determinant identity in higher dimension, 2014. arXiv:1405.0993.

49 Richard Zippel. Probabilistic algorithms for sparse polynomials. In *Symbolic and algebraic computation (EUROSAM '79, Internat. Sympos., Marseille, 1979)*, volume 72 of *Lecture Notes in Comput. Sci.*, pages 216–226. Springer, Berlin-New York, 1979.

Revision Notice

This is a revised version of the eponymous paper appeared in the proceedings of APPROX/RANDOM 2020 (LIPIcs, volume 176, https://www.dagstuhl.de/dagpub/978-3-95977-164-1, published in August, 2020), in which the proof of Theorem 6 is corrected. The proof in the previous version did not work, because the tester map constructed in Step 3, was incorrectly applied to $\alpha \in \mathbb{F}_{q^t}$, to get the points $\ell_1(\alpha), \ldots, \ell_\nu(\alpha) \in \mathbb{F}_q$, on which $f(x^{b_1}, x^{b_2}, \ldots, x^{b_n})$ was finally evaluated. In this revised version, the tester map is applied on $(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_{q^t}^n$ instead, to get the points $\ell_1(\alpha^{b_1}, \ldots, \alpha^{b_n}), \ldots, \ell_\nu(\alpha^{b_1}, \ldots, \alpha^{b_n}) \in \mathbb{F}_q^n$, on which $f(x_1, \ldots, x_n)$ is finally evaluated.

Dagstuhl Publishing – August 19, 2020.

Bounds for List-Decoding and List-Recovery of Random Linear Codes

Venkatesan Guruswami
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
venkatg@cs.cmu.edu

Ray Li
Department of Computer Science, Stanford University, CA, USA
rayyli@stanford.edu

Jonathan Mosheiff
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
jmosheif@cs.cmu.edu

Nicolas Resch
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
nresch@cs.cmu.edu

Shashwat Silas
Department of Computer Science, Stanford University, CA, USA
silas@stanford.edu

Mary Wootters
Department of Computer Science, Stanford University, CA, USA
marykw@stanford.edu

Abstract

A family of error-correcting codes is list-decodable from error fraction p if, for every code in the family, the number of codewords in any Hamming ball of fractional radius p is less than some integer L that is independent of the code length. It is said to be list-recoverable for input list size ℓ if for every sufficiently large subset of codewords (of size L or more), there is a coordinate where the codewords take more than ℓ values. The parameter L is said to be the "list size" in either case. The capacity, i.e., the largest possible rate for these notions as the list size $L \to \infty$, is known to be $1 - h_q(p)$ for list-decoding, and $1 - \log_q \ell$ for list-recovery, where q is the alphabet size of the code family.

In this work, we study the list size of *random linear codes* for both list-decoding and list-recovery as the rate approaches capacity. We show the following claims hold with high probability over the choice of the code (below q is the alphabet size, and $\varepsilon > 0$ is the gap to capacity).

- A random linear code of rate $1 - \log_q(\ell) - \varepsilon$ requires list size $L \geq \ell^{\Omega(1/\varepsilon)}$ for list-recovery from input list size ℓ. This is surprisingly in contrast to completely random codes, where $L = O(\ell/\varepsilon)$ suffices w.h.p.
- A random linear code of rate $1 - h_q(p) - \varepsilon$ requires list size $L \geq \lfloor h_q(p)/\varepsilon + 0.99 \rfloor$ for list-decoding from error fraction p, when ε is sufficiently small.
- A random *binary* linear code of rate $1 - h_2(p) - \varepsilon$ is list-decodable from *average* error fraction p with list size with $L \leq \lfloor h_2(p)/\varepsilon \rfloor + 2$. (The average error version measures the average Hamming distance of the codewords from the center of the Hamming ball, instead of the maximum distance as in list-decoding.)

The second and third results together precisely pin down the list sizes for binary random linear codes for both list-decoding and average-radius list-decoding to three possible values.

Our lower bounds follow by exhibiting an explicit subset of codewords so that this subset – or some symbol-wise permutation of it – lies in a random linear code with high probability. This uses a recent characterization of (Mosheiff, Resch, Ron-Zewi, Silas, Wootters, 2019) of configurations of codewords that are contained in random linear codes. Our upper bound follows from a refinement of the techniques of (Guruswami, Håstad, Sudan, Zuckerman, 2002) and strengthens a previous result of (Li, Wootters, 2018), which applied to list-decoding rather than average-radius list-decoding.



© Venkatesan Guruswami, Ray Li, Jonathan Mosheiff, Nicolas Resch, Shashwat Silas, and Mary Wootters;
licensed under Creative Commons License CC-BY

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 9; pp. 9:1–9:21



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

2012 ACM Subject Classification Mathematics of computing → Coding theory

Keywords and phrases list-decoding, list-recovery, random linear codes, coding theory

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.9

Category RANDOM

Related Version A full version of the paper is available at `https://arxiv.org/abs/2004.13247`.

Funding *Venkatesan Guruswami*: Partially funded by NSF grants CCF-1563742 and CCF-1814603.
Ray Li: Partially funded by NSF-CAREER grant CCF-1844628, NSF-BSF grant CCF-1814629, a Sloan Research Fellowship and NSF-GRFP grant DGE-1656518.
Jonathan Mosheiff: Partially funded by NSF grants CCF-1563742 and CCF-1814603.
Nicolas Resch: Partially funded by NSF grants CCF-1563742, CCF-1814603, CCF-1527110, CCF-1618280, CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808 and a Sloan Research Fellowship.
Shashwat Silas: Partially funded by NSF-CAREER grant CCF-1844628, NSF-BSF grant CCF-1814629, a Sloan Research Fellowship and a Google Graduate Fellowship.
Mary Wootters: Partially funded by NSF-CAREER grant CCF-1844628, NSF-BSF grant CCF-1814629 and a Sloan Research Fellowship.

1 Introduction

In coding theory, one is interested in the combinatorial properties of sets $\mathcal{C} \subseteq \mathbb{F}_q^n$.[1] Such a set $\mathcal{C}$ is called a *code* of *length* n over the *alphabet* $\mathbb{F}_q$, and the elements $c \in \mathcal{C}$ are called *codewords*.

List-decoding, introduced by Elias and Wozencraft in the 1950's [10, 44], is such a combinatorial property. For $p \in [0,1]$ and integer $L \geq 1$, we say that a code $\mathcal{C} \subseteq \mathbb{F}_q^n$ is (p, L)-*list-decodable* if, for all $z \in \mathbb{F}_q^n$,

$$|\{c \in \mathcal{C} : \delta(c, z) \leq p\}| < L,$$

where $\delta(x, y) = \frac{1}{n}|\{i : x_i \neq y_i\}|$ denotes relative Hamming distance. That is, $\mathcal{C}$ is list-decodable if not too many codewords of $\mathcal{C}$ live in any small enough Hamming ball. In this paper, we are interested in the trade-offs between p, L, and the rate of the code $\mathcal{C}$. The *rate* R of $\mathcal{C}$ is defined as $R = \frac{\log_q |\mathcal{C}|}{n}$. The rate lies in the interval $[0, 1]$, and larger is better.

Variations of list-decoding

In this work, we consider standard list-decoding along with two variations.

The first variation is a strengthening of list-decoding known as *average-radius list-decoding*. A code $\mathcal{C}$ is (p, L)-*average-radius list-decodable* if for any set $\Lambda \subseteq \mathcal{C}$ of size L and $z \in \mathbb{F}_q^n$,

$$\frac{1}{L}\sum_{c \in \Lambda} \delta(c, z) \geq p.$$

[1] Here and throughout the paper, $\mathbb{F}_q$ denotes the finite field with q elements. In this we work only consider linear codes, so we always assume that the alphabet is a finite field.

It is not hard to see that (p, L)-average-radius list-decodability implies (p, L)-list-decodability, and this stronger formulation has led to stronger lower bounds than are achievable otherwise [22]. In addition to stronger lower bounds, average-radius list-decoding – essentially replacing a maximum with an average in the definition of list-decoding – is a natural concept, and it has helped establish connections between list-decoding and compressed sensing [7].

The second variation, known as *list-recovery,* is a version where the "noise" is replaced by uncertainty about each symbol of the received word z. Formally, we say that a code $\mathcal{C}$ is (ℓ, L)-*list-recoverable* if for any sets $S_1, \ldots, S_n \subseteq \mathbb{F}_q$ with $|S_i| \leq \ell$ for all i,

$$|\{c \in \mathcal{C} : c_i \in S_i \, \forall i\}| < L.$$

List-recovery was originally used as a stepping-stone to list-decoding and unique-decoding (e.g., [17, 18, 19, 20]) but it has since become a useful primitive in its own right, with applications beyond coding theory [31, 38, 13, 28, 8].

Pinning down the output list size

We are motivated by the problem of pinning down the output list size L for (average-radius) list-decoding and for list-recovery. For all three of these problems, given q and p (respectively, q and ℓ), there exists an *optimal rate*, denoted R^*. Namely, R^* is the largest rate so that, for any $\varepsilon > 0$, there are q-ary codes of rate $R^* - \varepsilon$ and arbitrarily large length, which are (p, L)-(average-weight)-list-decodable (resp. (ℓ, L)-list-recoverable), for some $L(q, p, \varepsilon)$ (resp. $L(q, \ell, \varepsilon)$). Importantly, L must not depend on the length of the code. The *list-decoding capacity theorem* gives the dependence of R^* on q and p (resp. q and ℓ): $R^* = 1 - h_q(p)$ for both standard and average-radius list-decoding, [11, 45] and $R^* = 1 - \log_q(\ell)$ for list-recovery (e.g., [42]).

We are interested in the trade-off between the list size L, the parameters p, q, ℓ of the problem, and this gap ε; we refer to ε as the *gap to capacity.* Pinning down the list size L is an important problem. For example, for many of the algorithmic applications within coding theory, the list size represents a bottleneck on the running time of an algorithm that must check each item in the list before pruning it down [20, 9, 26, 27, 21]. For applications in pseudorandomness, for example to expanders or extractors, the list size corresponds to the expansion or to the amount of entropy in the input, respectively, and it is of interest to precisely pin down these quantities.

We make progress on pinning down the output list sizes for the case of *random linear codes.* A random linear code is a uniformly random subspace of $\mathbb{F}_q^n$ of certain dimension. The list-decodability of random linear codes has been well studied for many reasons [45, 15, 7, 43, 40, 42, 34]. First, it is a natural mathematical question that studies the interplay between two fundamental notions in $\mathbb{F}_q^n$: subspaces and Hamming balls. Second, there are constructions of codes which use random linear codes (and their list-decodability) as a building block [20, 24, 30, 29], and improvements in the parameters of random linear codes will lead to improvements in these constructions as well. Third, random linear codes can be seen as one way to partially derandomize completely random codes; this is especially motivating in the binary (or fixed alphabet) case, where we do not know of any explicit constructions of optimally list-decodable codes, linear or otherwise.

1.1 Contributions

Our main results are improved bounds on the list size of random linear codes. We defer the formal theorem statements until after we have set up notation, but we informally summarize our results here. Below, we consider codes of *rate* $R^* - \varepsilon$, where as above we use R^* to denote best achievable rate for each particular problem.

(1) **Lower bound on the list size for list-recovery of random linear codes.** We show that if a random linear code of rate $R^* - \varepsilon$ is list-recoverable with high probability with input list sizes ℓ and output list size L, then we must have $L = \ell^{\Omega(1/\varepsilon)}$. This is in contrast to completely random codes, for which the output list size is $L = O(\ell/\varepsilon)$ with high probability.
This gap between random linear codes and completely random codes demonstrates that in some sense zero-error list-recovery behaves more like *erasure-list-decoding* [14] than it does like list-decoding with errors. Such a gap is present between general and linear codes in erasure list-decoding, but as we see below, there is no such gap for list-decoding from errors.
Our result extends to the setting of list-recovery with erasures as well. The formal theorem statement and proof can be found in Section 3.

(2) **Better lower bounds on the list size for list-decoding random linear codes.** We show that if a q-ary random linear code of rate $R^* - \varepsilon$ is list-decodable with high probability up to radius p with an output list size of L, then we must have $L \geq \left\lfloor \frac{h_q(p)}{\varepsilon} + 0.99 \right\rfloor$. By [34], this result is tight for list-decoding of binary random linear codes up to a small additive factor. As an immediate corollary, $L \geq \left\lfloor \frac{h_q(p)}{\varepsilon} + 0.99 \right\rfloor$ for *average-radius* list-decoding of random linear codes as well, and, as we will see below, this is also tight for binary random linear codes, up to a small additive factor. We conjecture that the leading constant $h_q(p)$ is also correct for $q > 2$.
Previous work [22] has established that $L = \Omega(1/\varepsilon)$, but to the best of our knowledge this is the first work that pins down the leading constant. In particular, [22] shows that, in the situation above, we have $L \geq c_{p,q}/\varepsilon$, where $c_{p,q}$ is a constant that goes to zero as p goes to $1 - 1/q$. In contrast, we show below that the leading constant is at least $h_q(p)$, which goes to 1 as p goes to $1 - 1/q$.
The formal theorem statement and proof can be found in Section 4.

(3) **Completely pinning down the list size for average-radius list-decoding of binary random linear codes.** We prove a new upper bound on the *average-radius* list-decodability of binary random linear codes, which matches our lower bound, even up to the leading constant. More precisely, we show that with high probability, a random binary linear code of rate $R^* - \varepsilon$ is *average-radius* list-decodable up to radius p with $L \leq \lfloor h_2(p)/\varepsilon \rfloor + 2$.
Such a bound was known for standard list-decoding [34], but our upper bound holds even for the stronger notion of average-radius list-decoding, and improves the additive constant by 1.[2] In particular, this shows that for both list-decoding and average-radius list-decoding of binary random linear codes, the best possible L is concentrated on at most three values: $\lfloor h_q(p)/\varepsilon \rfloor + 2$, $\lfloor h_q(p)/\varepsilon \rfloor + 1$ and $\lfloor h(p)/\varepsilon + 0.99 \rfloor$. This tight concentration demonstrates the sharpness of our upper and lower bound techniques.
The formal theorem statement and proof can be found in Section A.

1.2 Overview of techniques

In this section, we give a brief overview of our techniques.

Lower bounds

To illustrate the techniques for our lower bounds, we warm up with a back-of-the-envelope calculation which suggests why the "right" answer for our result (1) above is $\ell^{\Omega(1/\varepsilon)}$.

[2] Under our definition of list-decoding, [34] show (p, L) list-decodability with $L = \lfloor h(p)/\varepsilon \rfloor + 3$.

Consider a random linear code $\mathcal{C} \subset \mathbb{F}_q^n$, of rate $R = 1 - \log_q(\ell) - \varepsilon$, where $\varepsilon \in (0, \frac{1}{2})$. That is, $\mathcal{C}$ is the kernel[3] of a uniformly random matrix sampled from $\mathbb{F}_q^{(1-R)n \times n}$. Suppose that ℓ is a prime power, and $q = \ell^t$ for some $t \geq 2$. Thus, $\mathbb{F}_\ell$ is a sub-field of $\mathbb{F}_q$. Let D be an integer slightly smaller than $\frac{1}{2\varepsilon}$ and let $L = \ell^D \geq \ell^{\Omega(1/\varepsilon)}$. We claim that $\mathcal{C}$ is unlikely to be (ℓ, L)-list-recoverable.

Given a matrix $M \in \mathbb{F}_q^{n \times D}$, we write $M \subseteq \mathcal{C}$ ("$\mathcal{C}$ contains M") to mean that each of the columns of M is a codeword in $\mathcal{C}$.

Let $\mathcal{M}$ denote the set of all full-rank matrices $M \in \mathbb{F}_q^{n \times D}$ that have the following property: for every row M_i of M there exists some $x_i \in \mathbb{F}_q^*$ such that all entries of M_i belong to the set $x_i \cdot \mathbb{F}_\ell$.

We will show that $\mathcal{M}$ is *bad* and *abundant*. By *bad* we mean that a linear code containing a matrix from $\mathcal{M}$ cannot be (ℓ, L)-list-recoverable. We say that $\mathcal{M}$ is *abundant* (for the rate R) if a random linear code of rate R is likely to contain at least one matrix from $\mathcal{M}$. Clearly, the combination of these properties means that $\mathcal{C}$ is unlikely to be (ℓ, L)-list-recoverable.

We first prove that $\mathcal{M}$ is bad. Assume that $\mathcal{C}$ contains some matrix $M \in \mathcal{M}$. By linearity of the code, $\mathcal{C}$ also contains every vector of the form Mu, for $u \in \mathbb{F}_q^D$. In particular, consider the set of vectors $B := \left\{Mu \mid u \in \mathbb{F}_\ell^D\right\} \subseteq \mathcal{C}$. Observe that $\mathcal{C}$ cannot be (ℓ, L)-list-recoverable, since B is a "bad list" for list-recoverability with these parameters: First, since M has full-rank, B is of cardinality $\ell^D = L$. Now, given $i \in [n]$, we need to show that there exists a subset $S_i \subseteq \mathbb{F}_q$ with $|S_i| = \ell$, such that $v_i \in S_i$ for all $v \in B$. We take S_i to be the set $x_i \cdot \mathbb{F}_\ell$, which contains all entries of the row M_i. For $j \in [D]$, write $M_{i,j} = x_i \cdot w_j$ ($w_j \in \mathbb{F}_\ell$), and let $v = Mu$ for some $u \in \mathbb{F}_\ell^D$. Then

$$v_i = \sum_{j=1}^{D} M_{i,j} u_j = x_i \cdot \left(\sum_{j=1}^{D} w_j \cdot u_j \right) \in x_i \cdot \mathbb{F}_\ell,$$

and we conclude that $\mathcal{M}$ is bad.

Showing that $\mathcal{M}$ is abundant is harder, and at this stage we only provide some intuition for this fact. Let us compute the expected number of matrices $M \in \mathcal{M}$ that are contained in $\mathcal{C}$. First, we estimate the cardinality of $\mathcal{M}$. One may generate a matrix in $\mathcal{M}$ by choosing each of its rows in an essentially independent fashion.[4] Choosing a row amounts to choosing one of $\frac{q-1}{\ell-1}$ sets of the form $x \cdot \mathbb{F}_\ell$ ($x \in \mathbb{F}_q^*$) and then taking each entry to be an element of that set. Accounting for multiple counting of the all-zero row, the number of possible rows is thus $\frac{q-1}{\ell-1} \cdot (\ell^D - 1) + 1$, which we approximate as $q \cdot \ell^{D-1}$. Thus, $|\mathcal{M}| \approx (q \cdot \ell^{D-1})^n$. Next, it is not hard to see that a random linear code contains a given matrix of rank r with probability $q^{-(1-R) \cdot r \cdot n}$. Consequently, for $M \in \mathcal{M}$, we have $\Pr[M \subseteq \mathcal{C}] = q^{-(1-R)Dn}$. Therefore,

$$\begin{aligned} \mathbb{E}\left|\{M \in \mathcal{M} : M \subseteq \mathcal{C}\}\right| &= |\mathcal{M}| q^{-(1-R)Dn} \approx \left(q \cdot \ell^{D-1}\right)^n \cdot q^{-(1-R)Dn} \\ &= \left(\ell^{-1} \cdot q^{1-\varepsilon D}\right)^n = \ell^{(-1+t(1-\varepsilon D))n}, \end{aligned}$$

where, the penultimate equality is due to substituting $1 - \log_q(\ell) - \varepsilon$ for R. Finally, since $t \geq 2$ and $D < \frac{1}{2\varepsilon}$, the right-hand side of the above is $\ell^{\Omega(n)}$. Thus, in expectation, $\mathcal{C}$ contains many "bad" lists for list-recovery.

---

[3] This is one of several natural models for a random linear code. Another possible model is taking a uniformly random subspace of dimension Rn. It is not hard to see that the total variation distance between these distributions is exponentially small. In particular, our model yields a code of dimension exactly Rn with probability $1 - \exp(-\Omega(n))$.

[4] We say "essentially" since the resulting matrix might not have full rank, but this happens with negligibly small probability.

Of course, this back-of-the-envelope calculation does *not* yield the result advertised above. It might be the case that, even though the expected number of $M \in \mathcal{M}$ so that $M \subseteq \mathcal{C}$ is large, the probability that such an M exists is still small. In fact, as [37] shows, there are simple examples where this does happen. Thus, proving that $\mathcal{M}$ is abundant requires more work.

A standard approach to show that $\mathcal{M}$ is abundant would be via the second-moment method. Recently, [37] gave a general theorem which encompasses second-moment calculations in this context. In particular, they showed that there is essentially only one reason that a set $\mathcal{M}$ might not be abundant: there exists some matrix $A \in \mathbb{F}_q^{D \times D'}$, such that the set $\{MA \mid M \in \mathcal{M}\}$ is small. If this occurs, we say that $\mathcal{M}$ is *implicitly rare*.[5] They used this result to study the list-decodability of random Low-Density Parity-Check codes, but we can use their result to do our second moment calculation. We show that our example of $\mathcal{M}$ above[6] is not implicitly rare, by showing that there is no such linear map A. This establishes that the back-of-the-envelope calculation is in fact correct. Appealing to the machinery of [37], rather than applying the second moment method from scratch, allows us to get tighter constants with slightly less work, and gives a more principled approach to our lower bounds; indeed, our result (2) follows the same outline.

The intuition for our second result (2) is similar: we give an example of a class $\mathcal{M}$ which is *bad* for list-decoding and *abundant.* We define $\mathcal{M}$ as follows: Let $u \in \mathbb{F}_q^D$ be a random vector with independent $\text{Bernoulli}_q(p)$ entries, namely, each entry is 0 with probability $1-p$, and chosen uniformly from $\mathbb{F}_q^*$ with probability p. Let x be uniformly sampled from $\mathbb{F}_q$. Let τ denote the distribution (over $\mathbb{F}_q^D$) of the random vector $u + x \cdot 1_D$. Finally, define $\mathcal{M}$ to be the set of all matrices $M \in \mathbb{F}_q^{n \times D}$, such that a uniformly sampled row of M has the distribution τ. As before, we show that $\mathcal{M}$ is abundant by showing that $\mathcal{M}$ is not implicitly rare and using the result of [37].

Upper bounds

Our argument for our upper bound result (3) closely follows that of [34], which itself builds on the argument of [16]. The argument imagines building the random linear code one dimension at a time and uses a potential function to show that, so long as we do not add too many dimensions, no ball intersects the code too much. We now provide an informal overview of our approach, specifically comparing and contrasting it with the arguments of Guruswami, Håstad, Sudan and Zuckerman [16]; and Li and Wootters [34].

Let $R = 1 - h(\rho) - \varepsilon$ and put $k := Rn$ (which we assume for exposition is an integer). Note that sampling a random linear code of rate R is the same as sampling $b_1, \ldots, b_k \in \mathbb{F}_2^n$ independently and uniformly at random and outputting $\text{span}\{b_1, \ldots, b_k\}$. Consider the "intermediate" codes $\mathcal{C}_i = \text{span}\{b_1, \ldots, b_i\}$; [34] (following [16]) define a potential function $S_{\mathcal{C}_i}$ and endeavor to show that $S_{\mathcal{C}_i}$ does not grow too quickly. The work [16] demonstrated that this holds in expectation; the work [34] improved their argument to show that it holds with high probability. In both cases the potential function is such that it is easy to show that, so long as $S_{\mathcal{C}}$ is $O(1)$, the code $\mathcal{C}$ is list-decodable.

The potential function in these works keeps track of the radius p list-size at each vector $x \in \mathbb{F}_2^n$, that is, the cardinalities $|\{c \in \mathcal{C}_i : \delta(x, c) \leq p\}|$ for $i = 1, \ldots, k$, and shows that so long as i is not too large all these cardinalities remain at most L. For average-radius list-decoding, we instead keep track of a sort of "weighted" list size, where codewords that

[5] The term "implicitly rare" is used by the first version of [37], available at `https://arxiv.org/abs/1909.06430v1`.

[6] More precisely, we study an example similar to this one; the example above was slightly tweaked to simplify the exposition for this back-of-the-envelope explanation.

are very close x are weighted more heavily. We can reuse much of the analysis from [34] to demonstrate that on the k-th step the potential function is still bounded by a constant (in fact, it is at most 2). The real novelty in our argument is a demonstration that, assuming this potential function is small, the code is indeed (p, L)-average-radius list-decodable. This step is more involved than the argument in [16, 34] to establish (p, L)-list-decodability.

1.3 Related work

We now highlight some related work. In what follows, ε is always the "gap-to-capacity", i.e., if the capacity for a particular problem is R^*, then the result concerns codes of rate $R^* - \varepsilon$.

Lower bounds for list sizes of arbitrary codes

It is known that a typical (i.e., uniformly random) list-decodable code of rate $R^* - \varepsilon$ has list size $L = \Theta(1/\varepsilon)$, and a natural question to ask is whether *every* code requires a list of size $L = \Omega(1/\varepsilon)$. Blinovsky ([5, 4]) showed that lists of size $\Omega_p(\log(1/\varepsilon))$ are necessary for list-decoding a code of rate $R^* - \varepsilon$. Later, Guruswami and Vadhan [25] considered the high-noise regime where $p = 1 - 1/q - \eta$ and showed that lists of size $\Omega_q(1/\eta^2)$ are necessary. Finally, Guruswami and Narayanan [22] showed that for *average-radius* list-decoding, the list size must be $\Omega_p(1/\sqrt{\varepsilon})$.

Existing lower bounds for random linear codes

For the special case of random linear codes, Guruswami and Narayanan [22] showed that lists of size $c_{p,q}/\varepsilon$ are necessary. The constant $c_{p,q}$ is not explicitly computed (and in fact relies on a constant from [15] which we discuss below), but one can deduce from the proof that if p tends to $1 - 1/q$ then $c_{p,q}$ will tend to 0. Their lower bound follows from a second moment method argument, i.e., they consider a certain random variable X whose positivity is equivalent to the failure of a random linear code to be list-decodable, and then show that $\mathrm{Var}(X) = o(\mathbb{E}[X])^2$. In this sense our approach is similar to theirs, because we rely on results from [37] which themselves are proved using a second moment method. However, we are able to get stronger results (in the sense that our leading constant does not decay as $p \to 1 - 1/q$, and moreover is optimal for binary codes). One of the reasons may be the notion of "implicit rareness" from [37], which provides a useful characterization of the lists contained in a random linear code.

The work [22] also established lower bounds on list-decoding random linear codes from *erasures*. While we do not discuss list-decoding from erasures in this work (except in the sense that erasure list-recovery is a generalization of list-decoding from erasures), this result is relevant to our work because [22] established an exponential lower bound of the form $L \geq \exp(\Omega(1/\varepsilon))$, in contrast to the list size $O(1/\varepsilon)$ that is attained by uniformly random codes. Thus, our results suggest that (zero-error) list-recovery behaves more like list-decoding from erasures than from errors, at least with respect to the list size of random (linear) codes.

Existing upper bounds for random linear codes

We now turn our attention to upper bounds on list sizes for random linear codes. A long line of works [45, 16, 15, 7, 43, 40, 42, 34] has studied this problem, and we highlight the most relevant results now. Zyablov and Pinsker [45] showed that random linear codes of rate $R^* - \varepsilon$ have lists of size at most $q^{1/\varepsilon}$.[7] Guruswami, Håstad, Sudan and Zuckerman [16] first

[7] For list-recovery with input lists of size ℓ, the argument of [45] shows that the list size is at most $q^{\ell/\varepsilon}$. Furthermore, their results for list-decoding also apply to average-radius list-decoding.

showed the existence of capacity-achieving binary linear codes with lists of size $O(1/\varepsilon)$. Li and Wootters [34] revisited their techniques and showed that in fact random linear codes of rate $R^* - \varepsilon$ have lists of size $O(1/\varepsilon)$ with high probability; moreover they computed the constant coefficient in the big-Oh notation. However, neither of these results apply to either average-radius list-decoding or to list-recovery. As discussed above in Section 1.2, our new upper bound is the result of an improvement of the techniques of [34], which extends their result to average-radius list-decoding.

As for larger alphabets, Guruswami, Håstad and Kopparty [15] showed that there exists a constant $C_{p,q}$ for which random linear codes are $(p, C_{p,q}/\varepsilon)$-list-decodable with high probability. Unfortunately, if p tends to $1 - 1/q$ then this constant tends to infinity. To address this, an ongoing line of works [7, 43, 41, 42] has studied the list-decodability of random linear codes in the "high-noise regime" where p is close to $1 - 1/q$; these results also apply to average-radius list-decodability. These results imply that for binary random linear codes, when $p = 1 - 1/q - \Theta(\sqrt{\varepsilon})$, random linear codes with rate $R^* - \varepsilon$ are average-radius list-decodable with list sizes $O(1/\varepsilon)$. However, the constant hiding in the big-Oh is not correct (in particular, the authors do not see how to make it smaller than 2). Moreover, these results only hold in a particular parameter regime for p and ε, and degrade as the alphabet size grows.

As for list-recovery, a result by Rudra and Wootters [42] guarantees that random linear codes with rate $R^* - \varepsilon$ over sufficiently large alphabets $\mathbb{F}_q$ have lists of sizes at most $(q\ell)^{O(\log(\ell)/\varepsilon)}$. To the best of our knowledge, no lower bounds were known.

Relevant results for other ensembles of codes

Lastly, we discuss some other results concerning other code ensembles. First of all, recent work of [37] shows that a random code from Gallager's ensemble of LDPC codes [12] achieves list-decoding capacity with high probability. More generally, they show that random LDPC codes have similar combinatorial properties to random linear codes, including list-decoding, average-radius list-decoding, and list-recovery. As part of their approach, they develop techniques to characterize the lists that appear in a random linear code with high probability, which we utilize for our work.

Finally, we note that there are no known explicit constructions of list-decodable codes of rate $R^* - \varepsilon$ which achieve a list size even of $O(1/\varepsilon)$. Over large alphabets, the best explicit constructions of capacity-achieving list-decodable or list-recoverable codes have list sizes at least $(1/\varepsilon)^{\Omega(1/\varepsilon)}$ (e.g., [33, 32]). Further, if one insists on *binary* codes, or even codes over alphabets of size independent of ε, we do not know of *any* explicit constructions of list-decodable codes with rate approaching R^*.

Two-point concentration

We showed that the optimal list size L of a random linear code is concentrated on at most three values for both list-decoding and average-radius list-decoding: $\lfloor h(p)/\varepsilon \rfloor + 2$, $\lfloor h(p)/\varepsilon \rfloor + 1$, and, if the value is different, $\lfloor h(p)/\varepsilon + 0.99 \rfloor$.

In [34, Theorem 2.5], it was also shown that the optimal list size of a *completely* random binary code is concentrated on two or three values for list-decoding. This type of concentration is also well studied in graph theory, where it is known that in Erdős-Rényi graphs, a number of graph parameters are concentrated on two values. Examples include the clique number (size of the largest clique) [36, 6], the chromatic number [35, 2, 1], and the diameter [39].

1.4 Discussion and open problems

In this work, we have made progress on pinning down the output list sizes for (average-radius) list-decoding and list-recovery for random linear codes. Before we continue with the technical portion of the paper, we highlight some open questions and future directions.

- We showed that random linear codes of rate $1 - h_q(p) - \varepsilon$ are not (p, L)-list-decodable for $L \sim \frac{h_q(p)}{\varepsilon}$. We conjecture this lower bound is tight, i.e. that random linear codes of rate $1 - h_q(p) - \varepsilon$ are (p, L)-(average-radius) list-decodable for $L = \frac{h_q(p)}{\varepsilon}(1 + o(1))$, where the $o(1) \to 0$ as $\varepsilon \to 0$. Our Theorem 14 (and earlier in [34] for list-decoding) shows it is true for $q = 2$, and we conjecture this is true for larger q.
- Our results show that list-decoding and average-radius list-decoding have essentially the same output list sizes over binary alphabets, for random linear codes. It would be interesting to extend this to larger alphabets, or even to more general families of codes. This is especially interesting given that there is an exponential gap in the best known lower bounds (on the list-size for arbitrary codes) between list-decoding and average-radius list-decoding for general codes.
- We have used different techniques for our upper and lower bounds. However, we think it is an interesting direction to use the characterization of [37] – which we used to prove our lower bounds – to prove upper bounds as well. This would entail showing that every sufficiently bad list is implicitly rare.
- Finally, we note that our lower bounds for list-recovery rely on the field $\mathbb{F}_q$ being an extension field (that is, $q = p^t$ for some $t > 1$). It is an interesting question whether or not an exponential lower bound also holds over prime fields. We note that other lower bounds on list-decoding and list-recovery for Reed-Solomon codes also apply only to extension fields [23, 3]; perhaps all of these bounds taken together are evidence that better list-decodability may be possible in general over prime fields.

1.5 Organization

In Section 2 we set up notation and formally state the results of [37] that we build on for our lower bounds. In Section 3 we state and sketch the proof of our lower bound on list-recovery of random linear codes. In Section 4 we state and sketch the proof of our lower bound on the list-decodability of random linear codes. In Appendix A we state and prove our upper bound on the list-decodability of random linear codes.

2 Preliminaries

In this section, we set notation and introduce the notions and results from [37] that we need for our lower bounds.

2.1 Notation

Unless otherwise specified, all logarithms are base 2. We use the notation $\exp(x)$ to mean e^x. For an integer a, we define $[a] := \{1, \ldots, a\}$. For a vector $x \in \mathbb{F}_q^A$ and $I \subset [A]$, we use $x_I \in \mathbb{F}_q^{|I|}$ to denote the vector $(x_i)_{i \in I}$ with coordinates from I in increasing order. We use $\mathbf{1}_D$ to denote the all ones vector of length D.

We use several notions from information theory. Define the q-ary entropy $h_q : [0,1] \to [0,1]$ by

$$h_q(x) \stackrel{\text{def}}{=} x \log_q(q-1) - x \log_q x - (1-x)\log_q(1-x) \tag{1}$$

We assume $q = 2$ if q is omitted from the subscript.

For a random variable X with domain $\mathcal{X}$, we use $H(X)$ to denote the *entropy* of X:

$$H(X) = -\sum_{x \in \mathcal{X}} \Pr_X(x) \log(\Pr_X(x)).$$

For a probability distribution τ, we may also use $H(\tau)$ to denote the entropy of a random variable with distribution τ.

Let X be a random variable supported on $\mathcal{X}$ and Y be a random variable supported on $\mathcal{Y}$. We define the *conditional entropy* of Y given X as

$$H(Y|X) = -\sum_{x \in \mathcal{X}, y \in \mathcal{Y}} p(x,y) \log \frac{p(x,y)}{p(x)}.$$

It is easy to check that $H(X) - H(X|Y) = H(Y) - H(Y|X)$ and we call this the *mutual information* $I(X;Y)$:

$$I(X;Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

For random variables X, Y, Z, we define the *conditional mutual information* $I(X;Y|Z)$ by

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) = H(Y|Z) - H(Y|X,Z)$$

Conditional entropy, mutual information, and conditional mutual information satisfy the *data processing inequality:* for any function f supported on the domain of Y, we have

$$H(X|f(Y)) \geq H(X|Y) \quad \text{and} \quad I(X;Y) \geq I(X;f(Y)) \quad \text{and} \quad I(X;Y|Z) \geq I(X;f(Y)|Z).$$

We also use *Fano's inequality*, which states that if X is a random variable supported on $\mathcal{X}$ and Y is a random variable supported on $\mathcal{Y}$, and if $f : \mathcal{Y} \to \mathcal{X}$ is a function and $p_{err} = \Pr_{X,Y}[f(Y) \neq X]$

$$H(X|Y) \leq h(p_{err}) + p_{err} \cdot \log(|\mathcal{X}| - 1)$$

We define

$$H_q(X) \stackrel{\text{def}}{=} \frac{H(X)}{\log q}, \qquad I_q(X;Y) = \frac{I(X;Y)}{\log q}.$$

and similarly for conditional entropy and conditional mutual information.

For a distribution τ on $\mathbb{F}_q^L$ and a matrix $A \in \mathbb{F}_q^{L' \times L}$, we define the distribution $A\tau$ on $\mathbb{F}_q^{L'}$ in the natural way by

$$\Pr_{A\tau}(x) = \sum_{\{y \in \mathbb{F}_q^L \,:\, Ay = x\}} \Pr_\tau(y),$$

namely, $A\tau$ is the distribution of the random vector Ay, where $y \sim \tau$.

We have defined list-decoding, average-radius list-decoding, and list-recovery in the introduction. We will in fact consider a more general version of list-recovery, which also tolerates erasures:

▶ **Definition 1** (List-recovery from erasures)**.** *A code $C \subset \mathbb{F}_q^n$ is (α, ℓ, L)*-list-recoverable from erasures *if the following holds. Let $S_1, \ldots, S_n \subset \mathbb{F}_q$ be lists so that $|S_i| \leq \ell$ for at least αn values of i. Then*

$$|\{c \in \mathcal{C} \,:\, \forall i \in [n], c_i \in S_i\}| < L.$$

We take $\alpha = 1$ if it is omitted.

2.2 Tools from [37]

As discussed in Section 1.2, for our lower bounds we use tools from the recent work [37]. We work with matrices $M \in \mathbb{F}_q^{n \times L}$ ($L \in \mathbb{N}$), where we view the columns of M as potential codewords in $\mathcal{C}$. We use the notation "$M \subseteq \mathcal{C}$" to mean that the columns of M are all contained in $\mathcal{C}$.

We group together sets of such matrices M according to their row distribution.

▶ **Definition 2** (τ_M, $\dim(\tau)$, $\mathcal{M}_{n,\tau}$)**.** *Given a matrix $M \in \mathbb{F}_q^{n \times L}$, the empirical row distribution defined by the rows of M over $\mathbb{F}_q^L$ is called the type τ_M of M. That is, τ_M is the distribution so that for $v \in \mathbb{F}_q^L$,*

$$\Pr_{\tau_M}(v) = \frac{|\{i \,:\, \textit{the } i\textit{'th row of } M \textit{ is equal to } v\}|}{n}.$$

For a distribution τ on $\mathbb{F}_q^L$, we use $\dim(\tau)$ to refer to $\dim(\mathrm{span}(\mathrm{supp}(\tau)))$. We use $\mathcal{M}_{n,\tau}$ to refer to the set of all matrices in $\mathbb{F}_q^{n \times L}$ which have empirical row distribution τ.

▶ Remark 3. We remark that for some distributions τ over $\mathbb{F}_q^L$, the set $\mathcal{M}_{n,\tau}$ may be empty due to $n \cdot \Pr_\tau(v)$ not being an integer. For such τ we can define $\mathcal{M}_{n,\tau}$ to consist of matrices M with either $\lfloor n \cdot \Pr_\tau(v) \rfloor$ or $\lceil n \cdot \Pr_\tau(v) \rceil$ copies of v. This has a negligible effect on the analysis as we always take n to be sufficiently large compared to other parameters, so for clarity of exposition we ignore this technicality.

Given $M \in \mathcal{M}_{n,\tau}$, note that $\mathcal{M}_{n,\tau}$ consists exactly of those matrices obtained by permuting the rows of M. In particular, since the *random linear code* model is invariant to such permutations, all of the matrices in $\mathcal{M}_{n,\tau}$ have the same probability of being contained in $\mathcal{C}$.

As discussed in Section 1.2, we prove a lower bound by exhibiting a distributions τ over $\mathbb{F}_q^L$ such that the corresponding set $\mathcal{M}_{n,\tau}$ is both *bad* and *abundant*. When $\mathcal{M}_{n,\tau}$ satisfies these properties, we say that τ itself is, respectively, *bad* and *abundant*.

The work [37] characterizes which distributions τ satisfy the *abundance* property, namely, which classes $\mathcal{M}_{n,\tau}$ are likely to have at least one of their elements appear (as a matrix) in a random linear code of a given rate. To motivate the definition below, suppose that the distribution τ has low entropy: $H_q(\tau) < \gamma \cdot \dim(\tau)$ for some $\gamma \in (0,1)$. This implies that the class $\mathcal{M}_{n,\tau}$ is not too big: more precisely, it is not hard to see that $|\mathcal{M}_{n,\tau}| \leq q^{H_q(\tau) \cdot n} \leq q^{\gamma \dim(\tau) n}$. Using a calculation like we did in Section 1.2, we see that, since $\mathcal{M}_{n,\tau}$ is not very large, it is unlikely for a random linear code of rate less than $1 - \gamma$ to contain a matrix from $\mathcal{M}_{n,\tau}$.

However, this is not the only reason that $\mathcal{M}_{n,\tau}$ might be unlikely to appear in a random linear code. As is shown in [37], it could also be because a random output of τ, subject to some linear transformation (perhaps to a space of smaller dimension), has low entropy. We call such distributions *implicitly rare:*

▶ **Definition 4** (γ-implicitly rare)**.** *We say that a distribution τ over $\mathbb{F}_q^L$ is* γ-implicitly rare *if there exists a full-rank linear transformation $A : \mathbb{F}_q^L \to \mathbb{F}_q^{L'}$ where $L' \leq L$ such that*

$$H_q(A\tau) < \gamma \cdot \dim(A\tau)$$

Observe that by taking A to be the identity map, we recover the case where τ itself has low entropy. Furthermore, note that every matrix in $\mathcal{M}_{n,A\tau}$ has all of its columns contained in the column-span of some matrix in $\mathcal{M}_{n,\tau}$. This implies that if no matrix in $\mathcal{M}_{n,A\tau}$ lies in a code, then no matrix in $\mathcal{M}_{n,\tau}$ lies in the code. Thus, abundance of the distribution $A\tau$ implies abundance of τ.

For an illustrative example of an implicitly rare distribution, we refer the reader to [37, Example 2.5]. Specifically, the example provides a case where for some full-rank matrix A, we have $H_q(A\tau)/\dim(A\tau) > H_q(\tau)/\dim(\tau)$.

Essentially, [37] shows that a row distribution τ is likely to appear in a random linear code (namely, τ satisfies the abundance property) if and only if it is *not* implicitly rare. The following theorem follows from Lemma 2.7 in [37].[8]

▶ **Theorem 5** (Follows from Lemma 2.7 in [37]). *Let $R \in (0,1)$ and fix $\eta > 0$. Let τ be a $(1-R-\eta)$-implicitly rare distribution over $\mathbb{F}_q^L$ ($L \in \mathbb{N}$), and let $\mathcal{C}$ be a random linear code of rate R. Then*

$$\Pr[\exists M \in \mathcal{M}_{n,\tau} : M \subseteq \mathcal{C}] \leq q^{-\eta n}$$

Conversely, suppose that τ is not $(1-R+\eta)$-implicitly rare. Then

$$\Pr[\exists M \in \mathcal{M}_{n,\tau} : M \subseteq \mathcal{C}] \geq 1 - n^{O_{L,q}(1)} \cdot q^{-\eta n} \ .$$

The first part of the theorem follows from a natural first-moment method argument, while the second part follows from the analogous second-moment argument. We emphasize that it is important that we allow arbitrary full-rank linear transformations $A : \mathbb{F}_q^L \to \mathbb{F}_q^{L'}$ in Definition 4: if we only allowed A to be the identity map, the second part of the theorem would be false.

3 Lower bounds for list-recovery with erasures

Our main result in this section is the following.

▶ **Theorem 6.** *Fix $0 \leq \rho < 1$. Fix a prime power $\ell \geq 2$ and an integer $t \geq 2$, and let $q = \ell^t$. Fix $0 < \varepsilon \leq \frac{1-\rho}{20t}$ and let $L = \ell^{\lceil \frac{1-\rho}{20\varepsilon} \rceil}$. For $n \in \mathbb{N}$, let $\mathcal{C} \subseteq \mathbb{F}_q^n$ denote a random linear code of rate $R := 1 - (\rho + (1-\rho)\log_q(\ell)) - \varepsilon$. Then the probability of $\mathcal{C}$ being $(1-\rho, \ell, L)$-erasure list-recoverable is at most $q^{-\Omega(n)}$.*

We will prove Theorem 6 below, after we build up the necessary building blocks. As discussed in Sections 1.2 and 2, to prove Theorem 6 we seek a distribution τ that is both is *bad* and *abundant*. That is, $\mathcal{C}$ should likely contains some matrix from $\mathcal{M}_{n,\tau}$, and the corresponding codewords should yield a counterexample to the list-recoverability of $\mathcal{C}$. We will describe our choice of τ in Definition 7; we will show that it is bad in Proposition 8; and finally we will show that it is not implicitly rare (and hence abundant by Theorem 5) in Lemma 9.

Our construction of the distribution τ follows similar lines to that in Section 1.2.

▶ **Definition 7** (The bad distribution τ for list-recovery lower bounds). *Fix ρ, ℓ, t, q as in Theorem 6. Let $D \geq t$ be a positive integer. Let $\mathbb{F}_\ell$ be a subfield of $\mathbb{F}_q$, where $q = \ell^t$ and $t \geq 2$. Let $\alpha_1, \ldots, \alpha_{(q-1)/(\ell-1)}$ be a set so that $\alpha_i \mathbb{F}_\ell^*$ are disjoint cosets of $\mathbb{F}_\ell^*$ partitioning $\mathbb{F}_q^*$. Let $L = \ell^D$. Let $G \in \mathbb{F}_\ell^{L \times D}$ be the matrix whose rows are all of the distinct elements of $\mathbb{F}_\ell^D$.*

Let σ be the distribution that with probability $1-\rho$ returns $\alpha_i u$ for (i,u) uniform in $\left\{1, \ldots, \frac{q-1}{\ell-1}\right\} \times \mathbb{F}_\ell^D$; and with probability ρ returns a uniformly random element of $\mathbb{F}_q^D$.

Let τ be the distribution given by Gv for $v \sim \sigma$.

[8] This is also given as Theorem 2.2 in the first version of [37], available at `https://arxiv.org/abs/1909.06430v1`.

To motivate this construction, consider first the $\rho = 0$ case. Now consider a matrix $M \in \mathbb{F}_q^{n \times L}$ that has row distribution given by τ. If we ignore the coefficients α_i, the columns of M span a D-dimensional subspace of $\mathbb{F}_\ell^n$. In particular, they are *bad,* in the sense that each coordinate of these codewords are contained in a list of size ℓ (namely, $\mathbb{F}_\ell$). Moreover, as soon as any D linearly independent columns of M are contained in $\mathcal{C}$, all of the columns of M are contained in $\mathcal{C}$; this suggests that it's relatively likely (compared to, say, a random matrix in $\mathbb{F}_q^{L \times n}$) that $M \subseteq \mathcal{C}$. These properties don't change when we multiply by the coefficients α_i: each coordinate is now contained in some list $\alpha_i \mathbb{F}_\ell$ rather than $\mathbb{F}_\ell$ (notice that the fact that the α_i are coset representatives means that all of these possible lists are disjoint, other than zero), and it's still just as likely that $M \subseteq \mathcal{C}$. However, by throwing these multiples α_i into the mix, we have increased the size of $\mathcal{M}_{n,\tau}$, making τ more *abundant.* In particular, note that, over all choices of (i, u), the value $\alpha_i u$ is distinct except when $u = 0$. Thus, τ has entropy close to the entropy of the uniform distribution on (i, u), so $H_q(\tau) \approx \log_q(q\ell^{D-1}) \approx D(\log_q(\ell) + \frac{1}{D})$. Using a similar idea, we can estimate the entropy of $A\tau$ for all matrices A, showing that τ is *not* $\log_q(\ell) + \frac{1}{10D}$ implicitly rare, implying that it is abundant.

To generalize to the $\rho > 0$ case, the construction essentially "frees" a ρ fraction of the coordinates relative to the $\rho = 0$ case. This further increases the size of $\mathcal{M}_{n,\tau}$ (making τ even more abundant), while still maintaining the badness property for list-recovery with a ρ fraction of erasures.

▶ **Proposition 8** (τ is bad). *Let τ be as in Definition 7. Let $\mathcal{C} \subseteq \mathbb{F}_q^n$, and let $M \in \mathcal{M}_{n,\tau}$. If $M \subseteq \mathcal{C}$, then $\mathcal{C}$ is not $(1-\rho, \ell, L)$-list-recoverable.*

Proof. Suppose that $M \subseteq \mathcal{C}$. Let $w_1, w_2, \ldots, w_n \in \mathbb{F}_q^L$ be the rows of M. It suffices to show that there are input lists $S_1, \ldots, S_n$ so that $w_j \in S_j^L$ for all $j \in [n]$, and so that for at least $(1-\rho)n$ values of $j \in [n]$, we have $|S_j| \leq \ell$. Recall that each row w_j of M is of the form Gv_j where a $(1-\rho)$ fraction of the v_j are of the form $\alpha_{i_j} \cdot u_j$ for $(i_j, u_j) \in [(q-1)/(\ell-1)] \times \mathbb{F}_\ell^D$, and a ρ fraction of the v_j are arbitrary vectors in $\mathbb{F}_q^D$.[9]

In the first case, set $S_j = \alpha_{i_j} \cdot \mathbb{F}_\ell$. Because the elements of G are all in $\mathbb{F}_\ell$, all the coordinates of $w_j = Gv_j = \alpha_{i_j} G u_j$ lie in S_j. Moreover by definition $|S_j| \leq \ell$. In the second case, set $S_j = \mathbb{F}_q$. By definition all the coordinates of $w_j \in \mathbb{F}_q^L$ lie in $S_j = \mathbb{F}_q$.

This completes the proof. ◀

Next, we claim that τ is not implicitly rare, which will imply that τ is abundant. Due to space constraints, the proof is omitted.

▶ **Lemma 9** (τ is abundant). *Let τ be as in Definition 7. Then τ is* not

$$\left(\rho + (1-\rho)\log_q(\ell) + \frac{1-\rho}{10D}\right)\text{-implicitly rare.}$$

We now show how to use Proposition 8 and Lemma 9 to prove Theorem 6.

Proof of Theorem 6. Let $0 < \varepsilon \leq \frac{1-\rho}{20t}$. Let τ be as in Definition 7, choosing $D = \lceil \frac{1-\rho}{20\varepsilon} \rceil$. By our choice of ε, we indeed have $D \geq t$. Lemma 9 shows that τ is *not* $(\rho + (1-\rho)\log_q(\ell) + \frac{1-\rho}{10D})$-implicitly rare. By choice of D, we have $\frac{1-\rho}{10D} > \varepsilon$. From Theorem 5 with $\eta = \frac{1-\rho}{10D} - \varepsilon$, we see that for any sufficiently large n, a random code of rate

[9] As per Remark 3, we may ignore the rounding issue that ρn may not be an integer. This is without loss of generality, as we may replace τ with a very similar distribution so that a $\lceil \rho n \rceil$ fraction of the v_j are arbitrary in $\mathbb{F}_q^D$, and adjust all parameters by a term that is $o(1)$ as $n \to \infty$.

$$\left(1-\left(\rho+(1-\rho)\log_q(\ell)+\frac{1-\rho}{10D}\right)\right)+\eta=1-\left(\rho+(1-\rho)\log_q(\ell)\right)-\varepsilon$$

contains ℓ^D codewords given by a matrix $M \in \mathcal{M}_{n,\tau}$ with probability at least $1-q^{\Omega(\varepsilon n)}$. By Proposition 8, if this occurs, then $\mathcal{C}$ is not $(1-\rho,\ell,L)$-list-recoverable. ◀

4 Lower bounds for list-decoding with errors

Our main theorem in this section is the following.

▶ **Theorem 10.** *Fix a prime power q, fix $p \in (0, 1-\frac{1}{q})$, and fix $\delta \in (0,1)$. There exists $\varepsilon_{p,q,\delta} > 0$ such that for all $\varepsilon \in (0, \varepsilon_{p,q,\delta})$ and n sufficiently large, a random linear code in $\mathbb{F}_q^n$ of rate $1-h_q(p)-\varepsilon$ is not $\left(p, \left\lfloor \frac{h_q(p)}{\varepsilon} - \delta \right\rfloor\right)$-list-decodable with probability $1-q^{-\Omega(n)}$.*

Our proof of Theorem 10 below follows the same outline as the proof of Theorem 6 above. We first define a bad distribution τ in Definition 11; then we will show that it is bad in Proposition 12; then we will show that it is not implicitly rare (and hence abundant by Theorem 5) in Lemma 13. Finally we will prove Theorem 10 from these pieces.

Below, we let $\text{Bernoulli}_q(p)$ be the distribution that returns $0 \in \mathbb{F}_q$ with probability $1-p$ and any other element of $\mathbb{F}_q$ with probability $\frac{p}{q-1}$.

▶ **Definition 11** (The bad distribution τ for list-decoding lower bounds)**.** *Let $p \in (0, 1-\frac{1}{q})$ and $\delta > 0$. Choose $L > 0$. Define the distribution τ on $\mathbb{F}_q^L$ as the distribution of the random vector $u + \alpha \mathbf{1}_L$, where $u \sim \text{Bernoulli}_q(p)^L$, and α is sampled independently and uniformly from $\mathbb{F}_q$.*

First, we observe that τ is indeed bad, in the sense that it provides a counter-example to list-decodability.

▶ **Proposition 12** (τ is bad)**.** *Let τ be as in Definition 11. Let $\mathcal{C} \subseteq \mathbb{F}_q^n$ and let $M \in \mathcal{M}_{n,\tau}$. If $M \subseteq \mathcal{C}$, then $\mathcal{C}$ is not (p,L)-list-decodable.*

Proof. Let $M \in \mathcal{M}_{n,\tau}$. We want to show that the columns of M all lie in a single ball of radius pn.

By definition of τ and $\mathcal{M}_{n,\tau}$, we may write the j-th row of M as $u^{(j)} + \alpha_j \mathbf{1}_L$, so that the empirical distribution of the pairs $(u^{(j)}, \alpha_j)_{1 \le j \le n}$ is $\text{Bernoulli}_q(p)^L \times \text{Uniform}(\mathbb{F}_q)$.[10]

For any $i \in [L]$, the number of $j \in [n]$ such that $M_{i,j} = u_i^{(j)} + \alpha_j \neq \alpha_j$ is exactly the number of times $u_i^{(j)} \neq 0$, which is pn, since $u_i^{(j)}$ is distributed as $\text{Bernoulli}_q(p)$. Thus, each column $M_{i,*}$ of M has distance at most pn from the word $(\alpha_1, \ldots, \alpha_n)$, so that any code containing M has L codewords in a ball of radius pn and hence is not (p,L)-list-decodable. ◀

Next, we claim that τ is appropriately implicitly rare for large enough L. Due to space constraints, the proof is omitted.

▶ **Lemma 13.** *Let $p \in (0, 1-\frac{1}{q})$ and let $\delta > 0$. There exists $L_{p,q,\delta}$ such that, for $L \ge L_{p,q,\delta}$, the distribution τ given in Definition 11 is not $\left(h_q(p) + \frac{h_q(p)}{L+\delta}\right)$-implicitly rare.*

[10] This is without loss of generality: if not, as per Remark 3, we can associate pairs with rows so that the empirical distribution is close to $\text{Bernoulli}_q(p)^L \times \text{Uniform}(\mathbb{F}_q)$ up to an additive factors that are $o(1)$ as $n \to \infty$. After adjusting parameters, this has a negligible effect on the analysis and final result.

We now show how to use Proposition 12 and Lemma 13 to prove Theorem 10.

Proof of Theorem 10. Let $L_{p,q,\delta/2}$ be as in Lemma 13 and choose $\varepsilon_{p,q,\delta} \stackrel{\text{def}}{=} \frac{h_q(p)}{L_{p,q,\delta/2}+1}$. Fix $\varepsilon < \varepsilon_{p,q,\delta}$. Let $L = \left\lfloor \frac{h_q(p)}{\varepsilon} - \delta \right\rfloor$. Let τ be as in Definition 11 with this choice of L. By Lemma 13, as $L \geq L_{p,q,\delta/2}$, τ is not $\left(h_q(p) + \frac{h_q(p)}{L+\delta/2}\right)$-implicitly rare. Thus, as $\varepsilon \leq \frac{h_q(p)}{L+\delta} < \frac{h_q(p)}{L+\delta/2}$, there is some constant $c_{p,q,\varepsilon} > 0$ so that τ is not $(h_q(p) + \varepsilon + c_{p,q,\varepsilon})$-implicitly rare.

Then Theorem 5 with $\eta = c_{p,q,\varepsilon}$ tells us that, for n sufficiently large, a random linear code of rate $1 - (h_q(p) + \varepsilon + c_{p,q,\varepsilon}) + c_{p,q,\varepsilon} = 1 - h_q(p) - \varepsilon$ contains L codewords given by some matrix $M \in \mathcal{M}_{n,\tau}$ with probability at least $1 - q^{-\Omega_{p,q,\varepsilon}(n)}$.

Finally, Proposition 12 implies that $\mathcal{C}$ is not (p, L)-list-decodable. Our choice of L proves the theorem. ◀

References

1 Dimitris Achlioptas and Assaf Naor. The two possible values of the chromatic number of a random graph. *Annals of Mathematics*, 162(3):1335–1351, 2005.

2 Noga Alon and Michael Krivelevich. The concentration of the chromatic number of random graphs. *Combinatorica*, 17(3):303–313, 1997.

3 Eli Ben-Sasson, Swastik Kopparty, and Jaikumar Radhakrishnan. Subspace polynomials and limits to list decoding of reed–solomon codes. *IEEE Transactions on Information Theory*, 56(1):113–120, 2009.

4 Vladimir M Blinovsky. Code bounds for multiple packings over a nonbinary finite alphabet. *Problems of Information Transmission*, 41(1):23–32, 2005.

5 Volodia M. Blinovsky. Bounds for codes in the case of list decoding of finite volume. *Problems of Information Transmission*, 22(1):7–19, 1986.

6 Béla Bollobás and Paul Erdös. Cliques in random graphs. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 80, pages 419–427. Cambridge University Press, 1976.

7 Mahdi Cheraghchi, Venkatesan Guruswami, and Ameya Velingker. Restricted isometry of fourier matrices and list decodability of random linear codes. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 432–442, 2013. doi:10.1137/1.9781611973105.31.

8 Dean Doron, Dana Moshkovitz, Justin Oh, and David Zuckerman. Nearly optimal pseudorandomness from hardness. Technical report, ECCC preprint TR19-099, 2019.

9 Zeev Dvir and Shachar Lovett. Subspace evasive sets. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 351–358. ACM, 2012.

10 Peter Elias. List decoding for noisy channels. *Wescon Convention Record, Part 2*, pages 94–104, 1957.

11 Peter Elias. Error-correcting codes for list decoding. *IEEE Transactions on Information Theory*, 37(1):5–12, 1991.

12 Robert G. Gallager. Low-density parity-check codes. *IRE Trans. Information Theory*, 8(1):21–28, 1962. doi:10.1109/TIT.1962.1057683.

13 Anna C Gilbert, Hung Q Ngo, Ely Porat, Atri Rudra, and Martin J Strauss. l2/l2-foreach sparse recovery with low risk. In *International Colloquium on Automata, Languages, and Programming*, pages 461–472. Springer, 2013.

14 Venkatesan Guruswami. List decoding from erasures: Bounds and code constructions. *IEEE Transactions on Information Theory*, 49(11):2826–2833, 2003.

15 Venkatesan Guruswami, Johan Håstad, and Swastik Kopparty. On the list-decodability of random linear codes. *IEEE Trans. Information Theory*, 57(2):718–725, 2011. doi:10.1109/TIT.2010.2095170.

16 Venkatesan Guruswami, Johan Håstad, Madhu Sudan, and David Zuckerman. Combinatorial bounds for list decoding. *IEEE Trans. Information Theory*, 48(5):1021–1034, 2002. doi:10.1109/18.995539.

17 Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001, Las Vegas, Nevada, USA*, pages 658–667, 2001. doi:10.1109/SFCS.2001.959942.

18 Venkatesan Guruswami and Piotr Indyk. Near-optimal linear-time codes for unique decoding and new list-decodable codes over smaller alphabets. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 812–821, 2002.

19 Venkatesan Guruswami and Piotr Indyk. Linear time encodable and list decodable codes. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 126–135, 2003.

20 Venkatesan Guruswami and Piotr Indyk. Efficiently decodable codes meeting gilbert-varshamov bound for low rates. In *SODA*, volume 4, pages 756–757. Citeseer, 2004.

21 Venkatesan Guruswami and Swastik Kopparty. Explicit subspace designs. *Combinatorica*, 36(2):161–185, 2016.

22 Venkatesan Guruswami and Srivatsan Narayanan. Combinatorial limitations of average-radius list-decoding. *IEEE Trans. Information Theory*, 60(10):5827–5842, 2014. doi:10.1109/TIT.2014.2343224.

23 Venkatesan Guruswami and Atri Rudra. Limits to list decoding reed-solomon codes. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 602–609, 2005.

24 Venkatesan Guruswami and Atri Rudra. Concatenated codes can achieve list-decoding capacity. *Electronic Colloquium on Computational Complexity (ECCC)*, 15(054), 2008. URL: http://eccc.hpi-web.de/eccc-reports/2008/TR08-054/index.html.

25 Venkatesan Guruswami and Salil P. Vadhan. A lower bound on list size for list decoding. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22–24, 2005, Proceedings*, pages 318–329, 2005. doi:10.1007/11538462_27.

26 Venkatesan Guruswami and Chaoping Xing. Folded codes from function field towers and improved optimal rate list decoding. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 339–350. ACM, 2012.

27 Venkatesan Guruswami and Chaoping Xing. List decoding Reed-Solomon, Algebraic-Geometric, and Gabidulin subcodes up to the Singleton bound. In *Proceedings of the forty-fifth annual ACM symposium on Theory of computing*, pages 843–852. ACM, 2013.

28 Iftach Haitner, Yuval Ishai, Eran Omri, and Ronen Shaltiel. Parallel hashing via list recoverability. In *Annual Cryptology Conference*, pages 173–190. Springer, 2015.

29 Brett Hemenway, Noga Ron-Zewi, and Mary Wootters. Local list recovery of high-rate tensor codes & applications. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 204–215. IEEE, 2017.

30 Brett Hemenway and Mary Wootters. Linear-time list recovery of high-rate expander codes. *Information and Computation*, 261:202–218, 2018.

31 Piotr Indyk, Hung Q Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1126–1142. SIAM, 2010.

32 Swastik Kopparty, Nicolas Resch, Noga Ron-Zewi, Shubhangi Saraf, and Shashwat Silas. On list recovery of high-rate tensor codes. *Electronic Colloquium on Computational Complexity (ECCC)*, 2019. URL: https://eccc.weizmann.ac.il/report/2019/080/.

33 Swastik Kopparty, Noga Ron-Zewi, Shubhangi Saraf, and Mary Wootters. Improved decoding of folded reed-solomon and multiplicity codes. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 212–223. IEEE, 2018.

34 Ray Li and Mary Wootters. Improved list-decodability of random linear binary codes. *arXiv preprint*, 2018. arXiv:1801.07839.

35 Tomasz Luczak. A note on the sharp concentration of the chromatic number of random graphs. *Combinatorica*, 11(3):295–297, 1991.

36 David W Matula. Employee party problem. In *Notices of the American Mathematical Society*, volume 19, pages A382–A382. Amer Mathematical Soc 201 Charles St, Providence, RI 02940-2213, 1972.

37 Jonathan Mosheiff, Nicolas Resch, Noga Ron-Zewi, Shashwat Silas, and Mary Wootters. Ldpc codes achieve list decoding capacity. *arXiv preprint*, 2019. arXiv:1909.06430.

38 Hung Q Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. In *International Colloquium on Automata, Languages, and Programming*, pages 557–568. Springer, 2011.

39 Oliver Riordan and Nicholas Wormald. The diameter of sparse random graphs. *Combinatorics, Probability and Computing*, 19(5-6):835–926, 2010.

40 Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 764–773. ACM, 2014.

41 Atri Rudra and Mary Wootters. It'll probably work out: improved list-decoding through random operations. *Electronic Colloquium on Computational Complexity (ECCC)*, 21:104, 2014. URL: http://eccc.hpi-web.de/report/2014/104.

42 Atri Rudra and Mary Wootters. Average-radius list-recovery of random linear codes. In *Proceedings of the 2018 ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2018.

43 Mary Wootters. On the list decodability of random linear codes with large error rates. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 853–860, 2013. doi:10.1145/2488608.2488716.

44 Jack Wozencraft. List decoding. *Quarter Progress Report*, 48:90–95, 1958.

45 Victor Vasilievich Zyablov and Mark Semenovich Pinsker. List concatenated decoding. *Problemy Peredachi Informatsii*, 17(4):29–33, 1981.

A Upper bounds for average-radius list-decoding over $\mathbb{F}_2$

In this section we prove the following theorem. Recall that we abbreviate $h(p) = h_2(p)$.

▶ **Theorem 14.** *Let $n \in \mathbb{N}$. Let $p \in (0, \frac{1}{2})$ and $R = 1 - h(p) - \varepsilon$, where $0 < \varepsilon < 1 - h(p)$. Let $L = \lfloor \frac{h(p)}{\varepsilon} + 2 \rfloor$. Then, a random linear code $\mathcal{C} \leq \mathbb{F}_2^n$ of rate R is (p, L)-average-radius list-decodable with probability $1 - 2^{-\Omega_{p,\varepsilon}(n)}$.*

Recall from the introduction that, following the techniques in [16] and [34], we imagine sampling independent and uniform vectors $b_1, \ldots, b_k$ and constructing the "intermediate" random linear codes $\mathcal{C}_i = \text{span}\{b_1, \ldots, b_i\}$. A potential function based argument is used to show that, with high probability, each of these intermediate codes is indeed (p, L)-average-radius list-decodable; in particular, this is true for $\mathcal{C}_k$.

Before discussing our potential function, we first briefly review the techniques of [16] and [34]; in particular, we describe the potential function they use. First, for a code $\mathcal{C}$ and a vector $x \in \mathbb{F}_2^n$, we define

$$L_{\mathcal{C}}(x) := |\{c \in \mathcal{C} : \delta(x, c) \leq p\}| \ .$$

In [16], the authors define

$$S_{\mathcal{C}} := \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} 2^{\varepsilon n L_{\mathcal{C}}(x)}$$

and observe that, for any $b_1, \ldots, b_i \in \mathbb{F}_2^n$,

$$\mathop{\mathbb{E}}_{b_{i+1} \sim \mathbb{F}_2^n} \left[S_{\mathcal{C}_i + \{0, b_{i+1}\}} \right] = S_{\mathcal{C}_i}^2 \ ,$$

where we recall $\mathcal{C}_i = \mathrm{span}\{b_1, \ldots, b_i\}$.[11] That is, the potential function squares in expectation, so the probabilistic method guarantees that we can choose some b_{i+1} for which $S_{\mathcal{C}_{i+1}} \le S_{\mathcal{C}_i}^2$. Thus, for some choice of $b_1, \ldots, b_k$, one has $S_{\mathcal{C}_k} \le (S_{\{0\}})^{2^k}$.

In [34], the definition of $S_{\mathcal{C}}$ is slightly modified:

$$S_{\mathcal{C}} := \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} 2^{\frac{\varepsilon n L_{\mathcal{C}}(x)}{1+\varepsilon}} \ .$$

This little bit of extra room allows to show that, in fact, with high probability over the choice of b_{i+1}, $S_{\mathcal{C}_i + \{0, b_{i+1}\}} \le S_{\mathcal{C}_i}^2$. By a union bound, it follows that with high probability, $S_{\mathcal{C}_k} \le (S_{\{0\}})^{2^k}$.

In either case, to conclude the proof, one observes the bound[12] $S_{\{0\}} \le 1 + 2^{-n(1-h(p)-\varepsilon)}$ and then uses

$$S_{\mathcal{C}_k} \le (S_{\{0\}})^{2^k} \le (1 + 2^{-n(1-h(p)-\varepsilon)})^{2^k} \le \exp 2^{k-n(1-h(p)-\varepsilon)} \le O(1)$$

for k chosen as above.

A.1 Alterations for average-radius list-decoding

While this argument analyzes the (absolute-radius) list-decodability of random linear codes very effectively, it is not immediately clear how to generalize the argument to study average-radius list-decodability. We now introduce the additional ideas we need to derive Theorem 14. We will fix a threshold parameter $\lambda \in (p, \frac{1}{2})$ for which $h(\lambda) < 1 - R = h(p) + \varepsilon$, to be determined later, and define

$$\eta \stackrel{\mathrm{def}}{=} 1 - R - h(\lambda).$$

We define the function $M_{R,\lambda} : [0,1] \to \mathbb{R}$ by

$$M_{R,\lambda}(\gamma) := \begin{cases} 1 - R - h(\gamma) & \text{if } \gamma < \lambda \\ 0 & \text{if } \gamma \ge \lambda \end{cases} .$$

▶ Remark 15. One can think of this quantity as a sort of "normalized entropy change" up to the threshold λ. Recalling that $1 - R = h(p) + \varepsilon$, if $\gamma < \lambda$, then

$$M_{R,\lambda}(\gamma) \approx \frac{1}{n} \left(h(p) - h(\lambda) \right) \approx \log \left(\frac{|B^n(p)|}{|B^n(\gamma)|} \right),$$

where $B^n(p)$ denotes the Hamming ball in $\mathbb{F}_2^n$ of radius p. Hence, $M_{R,\lambda}(\gamma)$ is something like a normalized "surprise" an observer would experience if they are expecting a random vector of weight $\le p$ and see a vector of weight $\le \gamma$.

[11] Here and throughout, for two subsets $A, B \subseteq \mathbb{F}_2^n$, we denote $A + B = \{a + b : a \in A, b \in B$. Thus, $\mathcal{C}_i + \{0, b_{i+1}\} = \mathcal{C}_{i+1}$.

[12] Actually, for the potential function in [34], one has $S_{\{0\}} \le 1 + 2^{-n(1-h(p)-\frac{\varepsilon}{1+\varepsilon})}$, but this difference does not matter for the conclusion.

For a linear code $\mathcal{C} \leq \mathbb{F}_2^n$ and $x \in \mathbb{F}_2^n$ we define

$$L_{\mathcal{C},R,\lambda}(x) := \sum_{y \in \mathcal{C}} M_{R,\lambda}(\delta(x,y)).$$

This is intuitively the "smoothed-out" list-size of x, where nearby codewords are weighted more heavily than far away codewords, and the weighting is given by the "entropy change" implied by the distance from x to y.

Next, we define

$$A_{\mathcal{C},R,\lambda}(x) := 2^{\frac{nL_{\mathcal{C},R,\lambda}(x)}{1+\eta}} \quad \text{and} \quad S_{\mathcal{C},R,\lambda} := \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} A_{\mathcal{C},R,\lambda}(x).$$

The quantity $S_{\mathcal{C},R,\lambda}$ is the potential function we analyze.

A.2 Proof of Theorem 14

In this subsection we prove Theorem 14. The quantities R and λ (and hence $\eta = 1-R-h(\lambda)$) will be fixed throughout – although the precise value of λ will be determined later – and so we will suppress their dependence and simply write $M(x)$, $L_{\mathcal{C}}(x)$, $A_{\mathcal{C}}(x)$ and $S_{\mathcal{C}}$.

First, we observe that the following analog of [34, Lemma 3.2] holds. The proof is a simple adaptation of theirs (which in turn follows [16]).

▶ **Lemma 16.** *For all $\mathcal{C} \leq \mathbb{F}_2^n$ and $b \in \mathbb{F}_2^n$,*

$$L_{\mathcal{C}+\{0,b\}}(x) \leq L_{\mathcal{C}}(x) + L_{\mathcal{C}}(x+b), \tag{2}$$

$$A_{\mathcal{C}+\{0,b\}}(x) \leq A_{\mathcal{C}}(x) \cdot A_{\mathcal{C}}(x+b). \tag{3}$$

Moreover, equality holds if and only if $b \notin \mathcal{C}$.

Next, we bound $S_{\{0\}}$. We have

$$S_{\{0\}} \leq 1 + 2^{-n} \sum_{\substack{x \in \mathbb{F}_2^n \\ \mathrm{wt}(x) \leq \lambda}} 2^{\frac{n \cdot (1-R-h(\mathrm{wt}(x)))}{1+\eta}} \leq 1 + \sum_{i=0}^{\lfloor \lambda n \rfloor} 2^{-n\left(1-h(i/n)-\frac{h(\lambda)+\eta-h(i/n)}{1+\eta}\right)}.$$

As this sum is dominated by its last term, we deduce

$$S_{\{0\}} \leq 1 + (\lambda n) 2^{-n\left(1-h(\lambda)-\frac{\eta}{1+\eta}\right)}. \tag{4}$$

From here, we can combine Lemma 16 and (4) to deduce

▶ **Lemma 17.** *Let $p \in (0, \frac{1}{2})$ and $R = 1 - h(p) - \varepsilon$ for $0 < \varepsilon < 1 - h(p)$. Let $\mathcal{C}_{Rn} \leq \mathbb{F}_2^n$ be a random linear code of rate R. Then $S_{\mathcal{C}_{Rn}} \leq 2$ with probability at least $1 - \exp(-\Omega_\eta(n))$.*

The proof of this lemma is completely analogous to that of [34, Lemma 3.3]. One only needs to be careful about the growth rate of $S_{\mathcal{C}}$. In particular, this proof crucially uses that η is positive. We again choose vectors $b_1, \ldots, b_{Rn}$ independently and uniformly at random. If $\mathcal{C}_i = \mathrm{span}\{b_1, \ldots, b_i\}$, we need "in expectation" that $S_{\mathcal{C}_i} \leq 1 + 2^{-\Omega(n)}$ for all i for the error bounds to succeed. As we expect the $o(1)$ term to roughly double, we need $2^{Rn} \cdot (S_{(0)} - 1) = 2^{-n(\eta - \frac{\eta}{1+\eta})} \leq 2^{-\Omega_\eta(n)}$.

Thus, in order to conclude Theorem 14, we are simply required to demonstrate that $S_{\mathcal{C}} \leq 2$ implies that $\mathcal{C}$ is (p, L)-average-radius list-decodable: this is the crux of our contribution. The main lemma we require is the following.

▶ **Lemma 18.** *Let $\mathcal{C} \leq \mathbb{F}_2^n$ be a linear code of rate R such that $S_{\mathcal{C}} \leq 2$. Then, for all $x \in \mathbb{F}_2^n$ and $D \subseteq \mathcal{C} \cap B(x, \lambda)$, it holds that*

$$\sum_{y \in D} h(\delta(x,y)) \geq (|D| - 1 - \eta)(1-R) - \frac{1+\eta}{n}.$$

Proof. First, observe that for any $x \in \mathbb{F}_2^n$,

$$L_{\mathcal{C}}(x) \geq \sum_{y \in D} ((1-R) - h(\delta(x,y))) = |D|(1-R) - \sum_{y \in D} h(\delta(x,y))$$

$$\text{so} \qquad \log A_{\mathcal{C}}(x) \geq n \frac{|D|(1-R) - \sum_{y \in D} h(\delta(x,y))}{1+\eta}. \tag{5}$$

Next, as $\delta(x,y) = \delta(x+z, y+z)$ for any $z \in \mathbb{F}_2^n$, we have, for any $x \in \mathbb{F}_2^n$ and $c \in \mathcal{C}$, that $L_{\mathcal{C}}(x) = L_{\mathcal{C}}(x+c)$ and hence $A_{\mathcal{C}}(x) = A_{\mathcal{C}}(x+c)$. Thus, $\max_{x \in \mathbb{F}_2^n} A_{\mathcal{C}}(x)$ is attained at at least $|\mathcal{C}|$ different values of x, so

$$S_{\mathcal{C}} = \frac{1}{2^n} \sum_{x \in \mathbb{F}_2^n} A_{\mathcal{C}}(x) \geq \frac{1}{2^n} \cdot |\mathcal{C}| \cdot \max_{x \in \mathbb{F}_2^n} A_{\mathcal{C}}(x) = 2^{-(1-R)n} \cdot \max_{x \in \mathbb{F}_2^n} A_{\mathcal{C}}(x).$$

Combining this with (5), we have, for any $x \in \mathbb{F}_2^n$,

$$\begin{aligned} 1 &\geq \log S_{\mathcal{C}} \geq -(1-R)n + \log\left(A_{\mathcal{C}}(x)\right) \\ &\geq n \cdot \left(-(1-R) + \frac{|D|(1-R) - \sum_{y \in D} h(\delta(x,y))}{1+\eta} \right) \\ &= n \cdot \frac{(|D| - 1 - \eta)(1-R) - \sum_{y \in D} h(\delta(x,y))}{1+\eta}. \end{aligned}$$

Rearranging yields the lemma. ◀

We may now conclude Theorem 14.

Proof of Theorem 14. Since $L = \lfloor \frac{h(p)}{\varepsilon} + 2 \rfloor > \frac{h(p)}{\varepsilon} + 1 = \frac{1-R}{\varepsilon}$, there exists $\eta > 0$ small enough so that for all sufficiently large n

$$L > \frac{1 - R + \eta + \frac{1+\eta}{n}}{\varepsilon - \eta}. \tag{6}$$

Thus, we define λ so that η (which we defined as $\eta = 1 - R - h(\lambda)$) satisfies (6). Let $\mathcal{C}$ be a random linear code of rate R. Due to Lemma 17, the conclusion of Lemma 18, holds with probability $1 - 2^{-\Omega_\eta(n)}$ for $\mathcal{C}$. It remains to show that, assuming n is sufficiently large, any code $\mathcal{C}$ satisfying the conclusion of Lemma 18 is (p, L)-average-radius list-decodable.

Let $x \in \mathbb{F}_2^n$ and $\Lambda \subseteq \mathcal{C}$ such that $|\Lambda| = L$; our goal is to show that, for all such x and Λ,

$$\frac{1}{L} \sum_{y \in \Lambda} \delta(x,y) > p. \tag{7}$$

Let

$$D = \{y \in \Lambda : \delta(x,y) \leq \lambda\}$$

and define

$$h^*(\alpha) = \begin{cases} h(\alpha) & \text{if } \alpha \leq \frac{1}{2} \\ 1 & \text{if } \alpha > \frac{1}{2} \end{cases}.$$

Now,

$$\sum_{y\in\Lambda} h^*(\delta(x,y)) \geq \sum_{y\in D} h(\delta(x,y)) + (L-|D|)h(\lambda) \tag{8}$$

$$\geq (|D|-1-\eta)(1-R) + (L-|D|)(1-R-\eta) - \frac{1+\eta}{n} \tag{9}$$

$$= (1-R)\cdot(L-1) - \eta\cdot(1-R) - \eta\cdot(L-|D|) - \frac{1+\eta}{n}$$

$$\geq (1-R)\cdot(L-1) - \eta\cdot(L+1) - \frac{1+\eta}{n}$$

$$= (1-R)L - (1-R) - \eta\cdot(L+1) - \frac{1+\eta}{n}$$

$$= Lh(p) - (1-R) - (L+1)\eta + L\varepsilon - \frac{1+\eta}{n} \tag{10}$$

$$= Lh(p) - (1-R+\eta) + L(\varepsilon-\eta) - \frac{1+\eta}{n}$$

$$> Lh(p). \tag{11}$$

Here, Inequality (8) holds because $h^*(\alpha) > h(\lambda)$ for all $\alpha > \lambda$; Inequality (9) is the conclusion of Lemma 18; Equality (10) follows from the fact that $R = 1 - h(p) - \varepsilon$; and Inequality (11) follows from (6). Thus, we deduce

$$\frac{1}{L}\sum_{y\in\Lambda} h^*(\delta(x,y)) > h(p). \tag{12}$$

Since h^* is concave,

$$h^*\left(\frac{1}{L}\sum_{y\in\Lambda}(\delta(x,y))\right) \geq \frac{1}{L}\sum_{y\in\Lambda} h^*(\delta(x,y)),$$

and so (7) follows from (12), the monotonicity of h^* and the fact that $h^*(p) = h(p)$. ◀

Is It Possible to Improve Yao's XOR Lemma Using Reductions That Exploit the Efficiency of Their Oracle?

Ronen Shaltiel
University of Haifa, Israel
https://cs.haifa.ac.il/~ronen/
ronen@cs.haifa.ac.il

Abstract

Yao's XOR lemma states that for every function $f : \{0,1\}^k \to \{0,1\}$, if f has hardness $2/3$ for $P/poly$ (meaning that for every circuit C in $P/poly$, $\Pr[C(X) = f(X)] \leq 2/3$ on a uniform input X), then the task of computing $f(X_1) \oplus \ldots \oplus f(X_t)$ for sufficiently large t has hardness $\frac{1}{2} + \epsilon$ for $P/poly$.

Known proofs of this lemma cannot achieve $\epsilon = \frac{1}{k^{\omega(1)}}$, and even for $\epsilon = \frac{1}{k}$, we do not know how to replace $P/poly$ by $\mathrm{AC}^0[\mathrm{PARITY}]$ (the class of constant depth circuits with the gates $\{\mathrm{AND},\mathrm{OR},\mathrm{NOT},\mathrm{PARITY}\}$ of unbounded fan-in).

Recently, Grinberg, Shaltiel and Viola (FOCS 2018) (building on a sequence of earlier works) showed that these limitations cannot be circumvented by *black-box reductions*. Namely, by reductions $\mathrm{Red}^{(\cdot)}$ that given oracle access to a function D that violates the conclusion of Yao's XOR lemma, implement a circuit that violates the assumption of Yao's XOR lemma.

There are a few known reductions in the related literature on worst-case to average case reductions that are *non-black box*. Specifically, the reductions of Gutfreund, Shaltiel and Ta Shma (Computational Complexity 2007) and Hirahara (FOCS 2018)) are "class reductions" that are only guaranteed to succeed when given oracle access to an oracle D from some efficient class of algorithms. These works seem to circumvent some black-box impossibility results.

In this paper we extend the previous limitations of Grinberg, Shaltiel and Viola to class reductions, giving evidence that class reductions cannot yield the desired improvements in Yao's XOR lemma. To the best of our knowledge, this is the first limitation on reductions for hardness amplification that applies to class reductions.

Our technique imitates the previous lower bounds for black-box reductions, replacing the inefficient oracle used in that proof, with an efficient one that is based on limited independence, and developing tools to deal with the technical difficulties that arise following this replacement.

2012 ACM Subject Classification Theory of computation → Circuit complexity

Keywords and phrases Yao's XOR lemma, Hardness amplification, black-box reductions

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.10

Category RANDOM

Funding *Ronen Shaltiel*: This research was supported by ISF grant 1628/17.

Acknowledgements We are grateful to Emanuele Viola for very helpful discussions, and to anonymous referees for comments and suggestions.

1 Introduction

Yao's XOR Lemma is a fundamental and celebrated result in complexity theory, that is extensively studied (from various aspects) and has found many applications. See [9] for a survey article.


© Ronen Shaltiel;
licensed under Creative Commons License CC-BY



▶ **Definition 1** (The XOR function). *Given a $f : \{0,1\}^k \to \{0,1\}$, and a number t, we define $f^{\oplus t} : \{0,1\}^{t \cdot k} \to \{0,1\}$, as follows: Given $y \in \{0,1\}^{t \cdot k}$, we view y as $(y_1, \ldots, y_t) \in (\{0,1\}^k)^t$, and define:*

$$f^{\oplus t}(y) = f(y_1) \oplus \ldots \oplus f(y_t)$$

Let U_k denote the uniform distribution on k bit strings. Loosely speaking, Yao's XOR lemma says that if a function f is "mildly hard one average" on input $X \leftarrow U_k$, then as t increases, computing $f^{\oplus t}$ on input $Y \leftarrow U_{tk}$, becomes "very hard on average".

▶ **Lemma 2** (Yao's XOR lemma, for poly-size circuits). *For every $f : \{0,1\}^k \to \{0,1\}$, and $t \leq \text{poly}(k)$ such that $t = \omega(\log k)$:*

If, for every $\text{poly}(k)$ size circuit C, $\Pr_{X \leftarrow U_k}[C(X) = f(X)] < \frac{2}{3}$,

Then, for every constant c, and every $\text{poly}(k)$ size circuit D, $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] < \frac{1}{2} + \frac{1}{k^c}$.[1]

One weakness of Yao's XOR lemma, is that it cannot be used to conclude a statement in which the "hardness on average" $\frac{1}{2} + \frac{1}{k^c}$ is replaced by $\frac{1}{2} + \frac{1}{k^{\omega(1)}}$. This holds, even if the number of repetitions t is increased from slightly larger than $\log k$ (as is the case in Lemma 2) to the maximal choice of $t = \text{poly}(k)$. Specifically, the following question is wide open:

▶ **Open problem 3** (Yao's XOR lemma for subpolynomial error?). *Is it true that for every $f : \{0,1\}^k \to \{0,1\}$, taking $t = \omega(\log k)$ (or even the maximal choice of $t = \text{poly}(k)$) it holds that:*

If, for every $\text{poly}(k)$ size circuit C, $\Pr_{X \leftarrow U_k}[C(X) = f(X)] < \frac{2}{3}$,

Then, for every $\text{poly}(k)$ size circuit D, $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] < \frac{1}{2} + \frac{1}{k^{\omega(1)}}$,

Another weakness of Yao's XOR Lemma is that known proofs fail to prove Yao's XOR lemma when replacing *P/poly* with many interesting constant depth circuit classes. An especially frustrating case is the class $\text{AC}^0[\text{PARITY}]$ of poly-size constant depth circuits over the gates $\{\text{AND,OR,NOT,PARITY}\}$ of unbounded fan-in. There are known lower bounds showing explicit functions that have hardness $\frac{2}{3}$ for $\text{AC}^0[\text{PARITY}]$ (or even $\frac{1}{2} - o(1)$ hardness for circuits of depth d and size $2^{k^{\Omega(1/d)}}$ [25, 28]) but lower bounds with hardness $\frac{1}{2} - \frac{1}{k}$ are unknown. This is a twenty five year old barrier that prevents us from "using the hybrid argument" when constructing pseudorandom generators for $\text{AC}^0[\text{PARITY}]$ (and related classes). This barrier limits the best known pseudorandom generators for $\text{AC}^0[\text{PARITY}]$ (and related classes) [6] to very poor seeds (See [6] for a discussion of this limitation). Specifically, the following question is wide open:

▶ **Open problem 4** (Yao's XOR lemma for constant depth circuits?). *Let G be the set of gates $\{\text{AND,OR,NOT,PARITY}\}$ of unbounded fan-in. Is it true that for every $f : \{0,1\}^k \to \{0,1\}$, taking $t = \omega(\log k)$ (or even the maximal choice of $t = \text{poly}(k)$) it holds that:*

If, for every $\text{poly}(k)$ size, constant-depth circuit C with gates in G, $\Pr_{X \leftarrow U_k}[C(X) = f(X)] < \frac{2}{3}$,

Then, for every $\text{poly}(k)$ size, constant-depth circuit D, with gates in G, $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] < \frac{1}{2} + \frac{1}{k}$,

[1] Naturally, in order to make this asymptotic statement precise, one needs to consider an infinite sequence of functions $\{f_k\}$ with growing input length (so that terms like "poly-size", "$\omega(\log k)$", and "constant" are well defined). We allow ourselves to be imprecise, as a more general, and quantitatively precise statement of Yao's XOR lemma is given below in Lemma 5.

1.1 Proofs of Yao's Lemma as (nonuniform) black-box reductions

Before discussing the best known proofs of Yao's XOR Lemma, let us state the lemma more precisely, in a more general and quantitative form. The next formulation is achieved using Impagliazzo's proof of Yao's XOR lemma [18, 9] together with the quantitative improvement of Klivans and Servedio [21] of Impagliazzo's hard-core lemma [18].

▶ **Lemma 5** (Yao's XOR lemma, General version). *There exist a constant c, and a polynomial p, such that for every $f : \{0,1\}^k \to \{0,1\}$, every $\epsilon, \delta > 0$, and every $t \geq c \cdot \frac{\log(1/\epsilon)}{\delta}$, setting $q = c \cdot \frac{\log(1/\delta)}{\epsilon^2}$, we have that:*

If, for every circuit C of size $s \geq p(t,k,q)$, $\Pr_{X \leftarrow U_k}[C(X) = f(X)] < 1 - \delta$,

Then, for every circuit D of size $s' = \frac{s}{q}$, $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] < \frac{1}{2} + \epsilon$.

The special case of Lemma 2 is obtained by taking s to be a polynomial in k, and $\delta = \frac{1}{3}$. In order to reduce the number of live parameters, we recommend that the reader focuses on these choices on a first reading. We point out that s' (which is the size of D) is smaller by a factor of $q = \Omega(\frac{1}{\epsilon^2})$, than s (which is the size of C). This implies that $s' \leq O(\epsilon^2 \cdot s)$, implying that $\epsilon \geq \Omega(\frac{1}{\sqrt{s}})$, and it is impossible to get $\epsilon < \frac{1}{s}$ with current proofs. (This is a more quantitative way to state the phenomenon in open problem 3).

All known proofs of Yao's XOR lemma work by *reduction*. That is, the proof shows a reduction that transforms a circuit D such that $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] \geq \frac{1}{2} + \epsilon$, into a circuit C such that $\Pr_{X \leftarrow U_k}[C(X) = f(X)] \geq 1 - \delta$. All known proofs are "nonuniform black-box reductions", meaning that they provide a reduction (namely an oracle circuit $\mathrm{Red}^{(\cdot)}(x, \alpha)$ where x is an input, and α is an "advice string") and the circuit C is obtained by $C(x) = \mathrm{Red}^D(x, \alpha)$ where α is an "nonuniform advice string" that may depend on f and D.[2] This is made precise in the next definition.

▶ **Definition 6** (Nonuniform black-box reduction for Yao's XOR lemma). *Let $\epsilon, \delta > 0$, and let k, t, a be integers. A $(\frac{1}{2} + \epsilon) \to (1 - \delta)$* black-box reduction for Yao's XOR lemma *(with input length k, t repetitions and advice length a) is an oracle circuit $\mathrm{Red}^{(\cdot)}(x, \alpha)$, where $x \in \{0,1\}^k$ and $\alpha \in \{0,1\}^a$, such that for every $f : \{0,1\}^k \to \{0,1\}$, the following holds:*

For every function $D : \{0,1\}^{tk} \to \{0,1\}$, such that $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] \geq \frac{1}{2} + \epsilon$,

there exists $\alpha \in \{0,1\}^a$, such that $\Pr_{X \leftarrow U_k}[\mathrm{Red}^D(X, \alpha) = f(X)] \geq 1 - \delta$.

The version of Yao's XOR lemma stated in Lemma 5, follows by showing the following reduction:

▶ **Lemma 7** (Known black-box reductions for Yao's XOR lemma). *There exist a constant c, and a polynomial p, such that for every integer k, every $\epsilon, \delta > 0$ such that $1 - \delta > \frac{1}{2} + \epsilon$, and every $t \geq c \cdot \frac{\log(1/\epsilon)}{\delta}$, there is a $(\frac{1}{2} + \epsilon) \to (1 - \delta)$ black-box reduction $\mathrm{Red}^{(\cdot)}(x, \alpha)$ for Yao's XOR lemma with input length k, t repetitions, and advice length a such that:*

- *R makes at most $q = c \cdot \frac{\log(1/\delta)}{\epsilon^2}$ queries to its oracle.*
- *R is an oracle circuit of size $r = p(t,k,q)$, (and in particular, $a \leq r$).*
- *R is an oracle circuit of constant depth d over the gates $\{AND, OR, NOT\}$ of unbounded fan-in and also uses one majority gate with fan-in q.*

[2] There is a formal connection between "black box hardness amplification" and list-decodable error correcting code [29], see for example the discussion in [27, 10]. Using this connection, it is known that black-box reductions for Yao's XOR lemma, must be nonuniform and use an advice string if $1 - \delta > \frac{1}{2} + \epsilon$ and $\epsilon < \frac{1}{4}$.

In order to understand the limitations that prevent known proofs from solving the aforementioned open problems, it is instructive to see how Lemma 5 follows from Lemma 7. Specifically, assume (for contradiction) that Lemma 5 does not hold and let D be a circuit of size s' that is violating the conclusion. By Lemma 7, there exists $\alpha \in \{0,1\}^a$, such that the circuit $C(x) = \mathrm{Red}^D(x,\alpha)$ computes $f(X)$ with success $1-\delta$ on $X \leftarrow U_k$. The size of C is bounded by $s = r + a + q \cdot s' \geq q \cdot s'$, and the obtained circuit C has depth at least d, and needs to compute majority on q bits. Summing up:

- The number of queries q made by the reduction is a lower bound on $\frac{s}{s'}$, meaning that $s' \leq \frac{s}{q}$ and as the known reductions have $q \geq \frac{1}{\epsilon^2}$ we cannot expect $\epsilon < \frac{1}{\sqrt{s}}$, and cannot solve open problem 3.
- The fact that the best known reductions requires a majority gate on $q \geq \frac{1}{\epsilon}$ inputs, means that we need to assume hardness against a class that can perform this computation. For $\epsilon = 1/k$, Razborov's lower bound [25] (see also [24]) shows that for every depth d', majority on k bits, cannot be computed by circuits of depth d' and size $2^{k^{\Omega(1/d')}}$ over the gates $\{\text{AND},\text{OR},\text{NOT},\text{PARITY}\}$, explaining why current reductions cannot solve open problem 4.

Limitations on black-box reductions

A sequence of works [32, 27, 12, 3, 2] culminating in [10], shows that known black-box reductions for Yao's XOR lemma must suffer from the limitations above: They require $q = \Omega(\frac{\log(1/\delta)}{\epsilon^2})$ queries, and require computing majority on input length $\Omega(\frac{1}{\epsilon})$.[3]

1.2 Class reductions

On a closer examination, black-box reductions seem to be an overkill for the task of proving Yao's XOR lemma. For proving Yao's XOR lemma, we don't need $\mathrm{Red}^{(\cdot)}$ to succeed given oracle access to *every* function D such that $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] \geq \frac{1}{2} + \epsilon$. It is sufficient that Red succeeds only given oracle access to functions D that are *efficiently computable* and belong to the class $\mathcal{D}$ of circuits with size s' (if we're in the setup of open problem 3) and size s' with constant depth (if we're in the setup of open problem 4).

This motivates a notion of *class reduction* (suggested for example in [14]) in which reductions are only required to succeed if given oracle access to a function D that belongs to some class $\mathcal{D}$ of "efficient circuits", and do not need to succeed when given oracle access to a function D that does not belong to $\mathcal{D}$. The definition of *class $\mathcal{D}$ reduction* below is identical to definition 6 with the single exception (that is underlined for emphasis) being that we only require the reduction to succeed when given oracle access to a function D from the class $\mathcal{D}$.

▶ **Definition 8** (Nonuniform class reduction for Yao's XOR lemma)**.** *Let $\epsilon, \delta > 0$ and let k, t, a be integers, and let $\mathcal{D}$ be some class of functions $D : \{0,1\}^{tk} \to \{0,1\}$. A $(\frac{1}{2}+\epsilon) \to (1-\delta)$* class $\mathcal{D}$ reduction for Yao's XOR lemma *(with input length k, t repetitions and advice length a) is an oracle circuit $\mathrm{Red}^{(\cdot)}(x,\alpha)$, where $x \in \{0,1\}^k$ and $\alpha \in \{0,1\}^a$, such that for every $f : \{0,1\}^k \to \{0,1\}$, the following holds:*

For every function $D : \{0,1\}^{tk} \to \{0,1\}$ in the class $\mathcal{D}$, such that $\Pr_{Y \leftarrow U_{tk}}[D(Y) = f^{\oplus t}(Y)] \geq \frac{1}{2} + \epsilon$,

there exists $\alpha \in \{0,1\}^a$, such that $\Pr_{X \leftarrow U_k}[\mathrm{Red}^D(X,\alpha) = f(X)] \geq 1 - \delta$.

[3] More formally, saying that $\mathrm{Red}^{(\cdot)}$ "requires computing majority on input length $\Omega(1/\epsilon)$" means that every such reduction $\mathrm{Red}^{(\cdot)}$ can be transformed into a circuit (with no oracle) of roughly the same size and depth as $\mathrm{Red}^{(\cdot)}$ for computing the majority function on inputs of length $\Omega(\frac{1}{\epsilon})$.

Note that a black-box reduction is a special case of a class reduction where $\mathcal{D}$ is the class of all boolean functions on tk bits. This raises the following questions:

1. Are there reductions in the literature that are class reductions but not black-box reductions?
2. Can class reductions circumvent the limitations on black-box reductions and solve open problem 3 or open problem 4?

The answer to the first question is affirmative in the sense that there are at least two examples that we are aware of, where a worst-case to average case amplification is proven by a reduction that is not black-box. Furthermore, in both cases, the reduction is a class reduction, and there is strong evidence that it cannot be made black-box.

The first example is a worst-case hardness to average case hardness tradeoff for SAT (with respect to a distribution sampled in quasipolynomial time) by Shaltiel, Gutfreund and Ta-Shma [13] (see also a related work [4, 11, 14]). The correctness of the reduction of [13] relies on the efficiency of the oracle and the term "class reduction" was suggested by Gutfreund and Ta-Shma [14]. It was also argued in [14] that limitations on black-box reductions proven by Bogdanov and Trevisan [5] can be extended to the scenario studied in [13], and show that if the class reduction of [13] (which is non-adaptive) could be made also black-box, then co-NP has nondeterministic circuits of quasipolynomial size.

Another example is Hirahara's recent worst-case to average case reductions for variants of MCSP and MINKT [17]. These reductions are non-black-box, in the sense that their correctness relies on the efficiency of their oracle. The aforementioned work of Bogdanov and Trevisan [5] shows that if these reductions can be made black-box, then these problems are in co-NP/poly, which is not known, and is false, if these problems are NP-complete. See [17] for an elaborate discussion of consequences of the existence of such black-box reductions.

1.3 Our results: limitations on class reductions for Yao's XOR lemma

In this paper we give evidence that the answer to the second question above is negative. We extend the aforementioned limitations of [10] on black-box reductions for Yao's XOR lemma to *class reductions* for any $\mathcal{D}$ of that contains circuits that have polynomial size and constant depth over the gates $\{\text{AND},\text{OR},\text{NOT},\text{PARITY}\}$ with unbounded fan-in. To the best of our knowledge, this is the first example of proving limitations on class reductions in this setup. Our results are stated formally in the next theorem.[4]

▶ **Theorem 9** (Limitations on class reductions for Yao's XOR lemma). *There exist constants $\delta_0 > 0$, $\nu > 0$, $d_0 > 1$ and a polynomial p such that: Let $\mathrm{Red}^{(\cdot)}(x,\alpha)$ be a $(\frac{1}{2}+\epsilon) \to (1-\delta)$ class $\mathcal{D}$ reduction for Yao's XOR lemma, with input length k, t repetitions and advice length a. Assume that:*

- *$\mathrm{Red}^{(\cdot)}$ is a size r oracle circuit, that makes at most q queries.*
- *The class $\mathcal{D}$ contains circuits of size $p(r)$ and depth d_0 over the gates $\{\text{AND},\text{OR},\text{NOT},\text{PARITY}\}$ of unbounded fan-in.*
- *$t, a, \frac{1}{\epsilon}, \frac{1}{\delta} \leq r \leq 2^{\nu \cdot k}$ and $\delta \leq \delta_0$.*

Then the following holds:

[4] We remark that any circuit of size r over the gates $\{\text{AND},\text{OR},\text{NOT},\text{PARITY}\}$ with unbounded fan-in, cannot use fan-in larger than r, and therefore can be simulated by a circuit of size $O(r^2)$ over the standard gates $\{\text{AND},\text{OR},\text{NOT}\}$ with bounded fan-in. This allows us to state our results in a way that captures both circuits of small depth (using gates with unbounded fan-in) and circuits that use the standard gates with bounded fan-in.

- $\text{Red}^{(\cdot)}$ *requires many queries, specifically:* $q = \Omega(\frac{\log(1/\delta)}{\epsilon^2})$.
- $\text{Red}^{(\cdot)}$ *requires majority, specifically: if in addition to the size restriction on* Red*, we also have that* $\text{Red}^{(\cdot)}$ *is an oracle circuit of depth* d *over a set of gates* G *that contains the gates* $\{$AND,OR,NOT,PARITY$\}$ *of unbounded fan-in, then the majority function over* $\Omega(\frac{1}{\epsilon})$ *bits can be computed by a circuit of size* $\text{poly}(r)$ *and depth* $O(d)$ *over the set of gates* G.

What kind of reductions are ruled out by this result?

Theorem 9 achieves exactly the same limitations on *class reductions* for Yao's XOR lemma as the limitations of [10] for *black-box reductions*. This is achieved for any class $\mathcal{D}$ that contains small circuits with constant depth over the gates $\{$AND,OR,NOT,PARITY$\}$ of unbounded fan-in, which is exactly the classes that come up if one wants to use class reductions to solve open problems 3 and 4. This shows that a class reduction cannot circumvent the known limitations on black-box reductions, and additional ideas are needed for solving open problems 3 and 4.

More specifically, for the purpose of solving open problem 3 one wants a $(\frac{1}{2} + \frac{1}{k^{\omega(1)}}) \to \frac{2}{3}$ class $\mathcal{D}$ reduction $\text{Red}^{(\cdot)}$ for Yao's XOR lemma, of size $\text{poly}(k)$, for the class $\mathcal{D}$ of all circuits of size $\text{poly}(k)$. This is ruled out by our lower bound on the number of queries. For the purpose of solving open problem 4 one wants a $(\frac{1}{2} + \frac{1}{k}) \to \frac{2}{3}$ class $\mathcal{D}$ reduction for Yao's XOR lemma, of size $\text{poly}(k)$ and constant depth over the gates $G = \{$AND,OR,NOT,PARITY$\}$, for the class $\mathcal{D}$ of all circuits of size $\text{poly}(k)$ and constant depth over the gates G. This is ruled out by our results that $\text{Red}^{(\cdot)}$ requires majority on inputs of length $\Omega(k)$, and Razborov's lower bound [25] showing that this cannot be done by circuits of depth d' and size $2^{k^{\Omega(1/d)}}$.

A potential weakness of our impossibility results, is that they require that the class $\mathcal{D}$ has circuits of size larger than the reduction (although it is allowed that $\mathcal{D}$ contains only circuits of smaller depth than the reduction). This allows a scenario in which for every polynomial p_1, there exists a larger polynomial p_2 such that there is a $(\frac{1}{2} + \frac{1}{k^{\omega(1)}}) \to \frac{2}{3}$ class $\mathcal{D}_{p_1}$ reduction Red_{p_1} of size $p_2(k)$ for the class $\mathcal{D}_{p_1}$ of circuits of size $p_1(k)$ (but not for the class of circuits of size $p(p_2(k))$ where p is the polynomial in Theorem 9). This is sufficient for solving open problem 3 and is not ruled out by our impossibility results.

An optimistic view is that this may point us to the kind of reductions we need to design, in order to solve the aforementioned open problems. We remark however that the aforementioned reduction by Hirahara [17] *does not* need to assume that the oracle is weaker than the reduction. (The reduction of [13] involves a more complicated scenario where there is also a third entity which is the samplable distribution, and so, it is arguable whether the reduction is more powerful than the oracle).

Theorem 9 is weaker than the results of [10] in the sense that the limitations of [10] apply not only to Yao's XOR lemma, but to *any* hardness amplification technique. More precisely, in the results of [10] one can replace $f^{\oplus t}$ by any other function f' over $n = 2^{o(k)}$ bits, with the same limitations. Our approach cannot give such a general result, but can be extended as follows:

Extension to any efficient hardness amplification construction

Our results immediately extend to any function f' over $n = 2^{o(k)}$ bits such that f' can be efficiently computed given access to f. More precisely, in Theorem 9 one can replace occurrences of the parameter t by n, and the theorem extends to any function f' such that there exists an oracle circuit $\text{Con}^{(\cdot)}$ of size $\text{poly}(r)$, and constant depth over the gates $\{$AND,OR,NOT,PARITY$\}$ of unbounded fan-in, such that $f' = \text{Con}^f$. Moreover, if we omit the restriction that $\text{Con}^{(\cdot)}$ has constant depth, then the theorem holds with respect to any class $\mathcal{D}$ that contains circuits of size $p(r)$.

Extension to hardness amplification based on sufficiently explicit linear codes

Using ideas from [32], our results also extend to the case of $\delta = 2^{-2k}$ (which captures worst-case to average case hardness amplification) for functions f' over $n = 2^{o(k)}$ bits, such that:

$$f'(y) = \sum_{x \in \{0,1\}^k} f(x) \cdot g(x,y),$$

where the sum is taken in the field $\mathbb{F}_2$, and $g : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}$ can be computed by circuits of size $\text{poly}(r)$ and depth d over the set G of gates.

This definition of f' corresponds to "hardness amplification by a linear map". More specifically, we can view g as a matrix A of order $2^k \times 2^n$ over $\mathbb{F}_2$ by $A_{x,y} = g(x,y)$, and view the truth tables of the functions f, f' as vectors over $\mathbb{F}_2$ of dimension 2^k, 2^n, respectively. In this interpretation, the definition of f' above, says that $f' = f \cdot A$, for a matrix A in which the entry $A_{x,y}$ can be efficiently computed given x, y.

Many worst-case to average-case hardness amplification results in the literature choose f' so that the truth table f' is obtained by applying an error correcting code on the truth table of f. (It was observed in [29] that there is a formal connection between black-box reductions for hardness amplification, and list-decodable error correcting codes, see for example [10] for a discussion). Typical choices of this error correcting code are linear codes (most commonly Reed-Muller concatenated with Hadamard) and our results apply to this scenario, with the weaker conclusion that $q = \Omega(\frac{\log r}{\epsilon^2})$, and the same conclusion for the case of majority.

Perspective

Limitations for black-box reductions are extensively studied in various settings in complexity theory and cryptography. In order to prove impossibility results on black-box reduction, it is sufficient to show the existence of an oracle D (*that does not need to be efficient*) on which the reduction cannot succeed.

Many impossibility results and limitations in the literature strongly utilize the ability to choose an oracle D that is not efficient. One notable example is the aforementioned results of Bogdanov and Trevisan [5] (that build on earlier work of Feigenbaum and Fortnow [7]). Indeed, this is why these limitations do not apply to class reductions like the aforementioned results [13, 17].

This work puts an emphasis on whether or not the oracle D that one designs when showing a black-box impossibility result, can be made efficient, and demonstrates that achieving this, has the additional benefit of also ruling out class reductions.

1.4 Some more related work

It is beyond the scope of this paper to survey the vast literature on Yao's XOR lemma and hardness amplification. The reader is referred to [9] for a survey on Yao's XOR lemma, and to [32, 27, 10] for detailed discussions on the more general problem hardness amplification.

A significant advantage of Yao's XOR lemma (over some other suggested methods of hardness amplification) is that the "construction" $f' = f^{\oplus t}$ can be computed very efficiently, when given oracle access to f. A line of work (that is orthogonal to studying the complexity of *reductions* for hardness amplification) is interested in the complexity of *constructions* yielding hardness amplification. This line of work is mostly interested in starting from worst-case hard functions (which correspond to $\delta < 2^{-k}$) and aims to design (or prove impossibility results for) efficient constructions $\text{Con}^{(\cdot)}$ for which one can prove that if f has hardness $1 - \delta$, then $f' = \text{Con}^f$ has hardness $\frac{1}{2} + \epsilon$. (See e.g., [30, 31, 23, 15] for further discussion).

In this orthogonal line of work, there are examples of *constructions* which are non-black-box, and utilize specific properties of the function f (for example that $f \in \text{NP}$ or that f is a low degree polynomial). This is a different form of "non-black-box" than the one studied in this paper, and it is interesting to combine the two orthogonal directions.

There is a large body of work on proving black-box impossibility results in cryptography. This study was initiated by Impagliazzo and Rufich [19] and is concerned both with issues that are related to black-box constructions and to black-box reductions. See for example the discussion in Reingold, Trevisan and Vadhan [26] for a taxonomy of various notions.

2 Technique and a road map for proof

Our results are obtained by carefully examining the argument of the black-box impossibility result of [10], replacing the inefficient oracle with an efficient one, and handling the technical difficulties arising from this modification.

In this section we survey our technique, and give a roadmap of the proof of Theorem 9. We assume the setup of Theorem 9. Specifically, let $\text{Red}^{(\cdot)}(x,\alpha)$ be a $(\frac{1}{2}+\epsilon) \to (1-\delta)$ class $\mathcal{D}$ reduction for Yao's XOR lemma, with input length k, t repetitions and advice length a, which satisfies the requirements of the theorem. Let r be the size of Red and let d (which is not necessarily a constant) be the depth of Red. Our goal is to show that $\text{Red}^{(\cdot)}$ requires many queries, and that $\text{Red}^{(\cdot)}$ requires majority.

Let $f : \{0,1\}^k \to \{0,1\}$ be some function that we choose later, and let $n = tk$ be the input length of $f^{\oplus t}$. We start by surveying the approach of of the previous papers (which only handle *black-box* reductions rather than *class* reductions).

2.1 The approach of [32, 27]

We first introduce the following notation.

▶ **Definition 10** (Random sequences/functions). *For a number $0 \le p \le 1$, and an integer q, we define a distribution Noise^q_p over $\{0,1\}^q$ which consists of q i.i.d. bit variables $\mathsf{Noise}^q_p(1),\ldots,\mathsf{Noise}^p_q(q)$ where each of them has probability p to be one. This notation also allows us to view Noise^q_p as a distribution over functions from $[q]$ to $\{0,1\}$.*

Following [32, 27] (and as done in later works [12, 10]) our plan is to show that a $(\frac{1}{2}+\epsilon) \to (1-\delta)$ reduction $\text{Red}^{(\cdot)}(x,\alpha)$ that makes q queries, can be transformed into a (distribution) over circuits $T : \{0,1\}^q \to \{0,1\}$ with no oracle (that have roughly the same size and depth as Red) that distinguishes $\mathsf{Noise}^q_{1/2-2\epsilon}$ from $\mathsf{Noise}^q_{1/2}$. We will prove the following lemma (which we call the "zoom lemma").

▶ **Lemma 11** (Zoom lemma). *Under the assumption of Theorem 9, for every $x \in \{0,1\}^k$, there exists a circuit T_x over q bits, with size $\text{poly}(r)$ and depth $O(d)$ over the set of gates G, such that:*

- $\Pr_{X \leftarrow U_k}[T_X(\mathsf{Noise}^q_{1/2-2\epsilon}) = 1] \ge 1 - 2\delta$.
- $\Pr_{X \leftarrow U_k}[T_X(\mathsf{Noise}^q_{1/2}) = 1] \le \frac{1}{2} + \frac{1}{200}$.

Shaltiel and Viola [27] (see also [22]) showed that Theorem 9 follows from Lemma 11. This is formally stated and explained in Section A.[5]

[5] On an intuitive level, the connection between the consequence of the zoom lemma and the consequence of Theorem 9 is that the "best way" to distinguish $\mathsf{Noise}^q_{1/2-2\epsilon}$ from $\mathsf{Noise}^q_{1/2}$ is to check whether the

In the remainder of this section, we prove Lemma 11 modulo some other lemmas and claims, that are stated in this section, and proven in later sections of the paper.

2.2 The oracle used for black-box reductions

Lemmas that are similar to the zoom lemma are at the heart of earlier results [27, 12, 10] on *black-box* reductions, and we would like to imitate the argument working with *class* reductions. Let us start by explaining the oracle used in previous works.

Specifically, let us set $N = 2^n$ and identify the set $[N]$ with the set $\{0,1\}^n$ (so that we can think of Noise_p^N as a function $\mathsf{Noise}_p^N : \{0,1\}^n \to \{0,1\}$). We consider the following two (distributions over) oracles $D : \{0,1\}^n \to \{0,1\}$.

- $D_{1/2-2\epsilon}(y) = f^{\oplus t}(y) \oplus \mathsf{Noise}_{1/2-2\epsilon}^N(y)$
- $D_{1/2}(y) = f^{\oplus t}(y) \oplus \mathsf{Noise}_{1/2}^N(y)$.

▶ **Definition 12.** *We say that a function $D : \{0,1\}^n \to \{0,1\}$ is* useful*, if there exists an $\alpha \in \{0,1\}^a$ such that $\Pr_{X \leftarrow U_k}[\mathrm{Red}^D(X,\alpha) = f^{\oplus t}(X)] \geq 1 - \delta$.*

In the oracle $D_{1/2}$, the noise $\mathsf{Noise}_{1/2}^N(y)$ is uniform and completely masks out the information in $f^{\oplus t}(y)$. Intuitively, this means that the oracle $D_{1/2}$ isn't useful for the reduction. On the other hand, a Chernoff bound shows that w.h.p. over choosing $h \leftarrow \mathsf{Noise}_{1/2-2\epsilon}^N$, we have that $|\{y \in \{0,1\}^n : h(y) = 1\}| \leq (\frac{1}{2} - \epsilon) \cdot N$. This gives that w.h.p. over choosing $D \leftarrow D_{1/2-2\epsilon}$, we have that $\Pr_{Y \leftarrow U_n}[D(Y) = f^{\oplus t}(Y)] \geq \frac{1}{2} + \epsilon$.

If Red is a $(\frac{1}{2} + \epsilon) \to (1 - \delta)$ *black-box* reduction, then by definition, this implies that every such good D is useful. The proof of [32, 27] then proceeds to transform a *black-box* reduction Red into the circuits T_x required from Lemma 11. We will elaborate on this argument shortly.

Our plan is to imitate this argument when Red is not necessarily a *black-box reduction*, and is only guaranteed to be a *class* reduction. Using this weaker assumption, we are not guaranteed that w.h.p. $D \leftarrow D_{1/2-2\epsilon}$ is useful. This is because we are not guaranteed that w.h.p. $D \leftarrow D_{1/2-2\epsilon}$ belongs to the class $\mathcal{D}$, and the reduction does not need to succeed if $D \notin \mathcal{D}$.

2.3 Using limited independence to obtain efficient oracles

We would like to make the oracle $D_{1/2-2\epsilon}$ efficiently computable by small circuits, so that it belongs to $\mathcal{D}$. This presents two difficulties:

1. $f^{\oplus t}$ is harder to compute than f (and f is assumed to be hard).
2. $\mathsf{Noise}_{1/2-2\epsilon}^N$ is a random function, and w.h.p. requires circuits of exponential size.

In order to circumvent the first problem we use an idea from [32] and will choose the function $f : \{0,1\}^k \to \{0,1\}$ in the following way:

▶ **Lemma 13.** *There exist constants c_1 such that for every constant c_2, there exists a function $f : \{0,1\}^k \to \{0,1\}$ such that:*

- *For every circuit $B : \{0,1\}^k \to \{0,1\}$ of size r^{c_2}, $\Pr_{X \leftarrow U_k}[B(X) = f(X)] \leq \frac{1}{2} + \frac{1}{200}$.*
- *f can be computed by a DNF of size $r^{c_1 \cdot c_2}$.*

fraction of ones is below or above $\frac{1}{2} - \epsilon$. This is similar in spirit to majority over inputs of length $\Omega(1/\epsilon)$, and it can be shown that majority on length $\Omega(1/\epsilon)$ can be reduced to this task. A Chernoff bound shows that $q = O(\frac{\log(1/\delta)}{\epsilon^2})$ is sufficient to distinguish between the two distributions with confidence $1 - \delta$, and it can be shown that such a confidence requires $q = \Omega(\frac{\log(1/\delta)}{\epsilon^2})$.

Proof. By a standard counting argument, there exists a constant c_1 such that for every constant c_2, setting $m = c_1 \cdot c_2 \cdot \log r$, there exists a function $g : \{0,1\}^m \to \{0,1\}$ such that for every circuit B of size $2^{m/c_1} = r^{c_2}$, $\Pr_{X \leftarrow U_m}[B(X) = g(X)] \leq \frac{1}{2} + \frac{1}{200}$. By choosing $\nu > 0$ to be sufficiently small as a function of c_2, we can get that $m \leq k$. The function $f : \{0,1\}^k \to \{0,1\}$ is the function that given $x \in \{0,1\}^k$ applies g on the first m bits of x. ◀

We choose f by the lemma, where c_2 is a constant that we choose later. With this choice we have that:

▶ **Corollary 14.** *The function $f^{\oplus t}$ can be computed by circuits of size* $\mathrm{poly}(r)$ *and constant depth over the gates* $\{$AND,OR,NOT,PARITY$\}$ *of unbounded fan-in.*

▶ Remark 15 (Replacing $f^{\oplus t}$ by a different target function f'). Corollary 14 is the only place in the proof where we use specific properties of $f^{\oplus t}$. The corollary holds for every function $f' : \{0,1\}^n \to \{0,1\}$ for which there exists an oracle circuit $\mathrm{Con}^{(\cdot)}$ of size $\mathrm{poly}(r)$ and constant depth over the gates $\{$AND,OR,NOT,PARITY$\}$ of unbounded fan-in, such that $f' = \mathrm{Con}^f$. This means that our results hold for every such function f'. Furthermore, if Con does not have constant depth, then Corollary 14 gives a size bound on f', and this is sufficient to show the lower bound on number of queries with respect to the class $\mathcal{D}$ of circuits of size $p(r)$.

Corollary 14 takes care of the first difficulty above. It says that $f^{\oplus t}$ can be computed by circuits in the class $\mathcal{D}$. We would like to replace $\mathsf{Noise}^N_{1/2-2\epsilon}$ by a (distribution) over efficient circuits in $\mathcal{D}$. Our approach is to replace $\mathsf{Noise}^N_{1/2-2\epsilon}$ (which consists of N independent bits) by a distribution which is ℓ-wise independent, for $\ell = \mathrm{poly}(r)$.

▶ **Definition 16** (ℓ-wise independence with bias p)**.** *A sequence $R_1, \ldots, R_N$ of bit random variables is ℓ-wise independent with bias p, if $R_1, \ldots, R_N$ are ℓ-wise independent, and for every $i \in [N]$, $\Pr[R_i = 1] = p$.*

We will rely on the following theorem by Gutfreund and Viola [16] (which is usually stated for $p = \frac{1}{2}$ but immediately extends to every rational $p = \frac{a}{b}$ as stated below:)

▶ **Theorem 17** ([16])**.** *Let $N = 2^n$. For every integers $\ell \leq N$ and $a \leq b$, setting $p = \frac{a}{b}$, there exists a distribution H^ℓ_p over circuits $h : \{0,1\}^n \to \{0,1\}$ of size* $\mathrm{poly}(n, \ell, b)$ *and depth $O(1)$ (over the gates* $\{$AND,OR,NOT,PARITY$\}$ *of unbounded fan-in) such that the distribution obtained by choosing $h \leftarrow H^\ell_p$ and considering $(h(1), \ldots, h(N))$, is ℓ-wise independent with bias p.*[6]

We will assume w.l.o.g. that $\frac{1}{\epsilon}$ is an integer, and set $\ell = p_0(r)$ for a polynomial p_0 that we will specify later. We define the following two (distributions over) oracles $D : \{0,1\}^n \to \{0,1\}$, in which we replace the independent bits of Noise^N by ℓ-wise independent bits:[7]

---

[6] We remark that the result of Gutfreund and Viola [16] is significantly stronger. More specifically, for our purposes it suffices that there is a family H of ℓ-wise independent hash functions $h : \{0,1\}^n \to \{0,1\}^b$, such that every h can be computed by the type of circuits claimed above. The result of Gutfreund and Viola gives a stronger bound on the size of H, and also shows that there is a *uniform* circuit that given the "index of h" and an input x, computes $h(x)$.

[7] Replacing fully independent oracles by limited independence oracles, and arguing that black-box procedures with few queries cannot tell the difference, is a common approach in proving black-box impossibility results, originating from the work of Goldreich and Krawczyk [8]. It should be noted that even when ignoring the issue of class reductions, and focusing on black-box reductions, we are considering reductions which are *nonuniform*. Nonuniform reductions get an advice string α that depends on the choice of the oracle. Loosely speaking, this may give them information about the "seed" used to generate the limited independence oracle. This creates technical difficulties that do not occur when reductions are uniform.

- $D^{\ell}_{1/2-2\epsilon}(y) = f^{\oplus t}(y) \oplus H^{\ell}_{1/2-2\epsilon}(y)$
- $D^{\ell}_{1/2}(y) = f^{\oplus t}(y) \oplus H^{\ell}_{1/2}(y)$.

We now have that every D in the support of $D^{\ell}_{1/2-2\epsilon}$ has size $r^{c_1 \cdot c_2} + \mathrm{poly}(n, \ell, 1/\epsilon)$ which can be bounded by $p(r)$ for some polynomial p. Furthermore, each such D has constant depth over the set of gates $\{\text{AND},\text{OR},\text{NOT},\text{PARITY}\}$. This gives that every such D is sufficiently efficient, and belongs to the class $\mathcal{D}$.

This will allow us to imitate the argument for black box reductions. Specifically, by Chebyshev's inequality, with probability at least $1 - \frac{1}{\epsilon^2 2^n} \geq \frac{1}{2}$ over choosing $h \leftarrow H^{\ell}_{1/2-2\epsilon}$, we have that $|\{y \in \{0,1\}^n : h(y) = 1\}| \leq (\frac{1}{2} - \epsilon) \cdot N$. This means that with probability at least half over choosing $D \leftarrow D^{\ell}_{1/2-2\epsilon}$, we have that $\Pr_{Y \leftarrow U_n}[D(Y) = f^{\oplus t}(Y)] \geq \frac{1}{2} + \epsilon$. As Red is a $(\frac{1}{2} + \epsilon) \to (1 - \delta)$ class $\mathcal{D}$ reduction, and every such good D belongs to $\mathcal{D}$, we get that:

▷ Claim 18. $\Pr_{h \leftarrow H^{\ell}_{1/2-2\epsilon}}[(f^{\oplus t} \oplus h) \text{ is useful}] \geq \frac{1}{2}$.

2.4 A more general fixed set lemma

We will now proceed in a similar manner to [27, 10]. Specifically, let Advice be a function that given a useful D, produces an advice string α such that $\Pr_{X \leftarrow U_k}[\mathrm{Red}^D(x, \alpha) = f^{\oplus t}(y)] \geq 1 - \delta$ (such an α exists by definition). For every $\alpha \in \{0,1\}^a$, let A_α be the event

$$A_\alpha = \{h : \{0,1\}^n \to \{0,1\} : (f^{\oplus t} \oplus h) \text{ is useful, and } \mathsf{Advice}(f^{\oplus t} \oplus h) = \alpha\}.$$

By averaging over the 2^a advice strings we obtain that:

▷ Claim 19. There exists $\alpha' \in \{0,1\}^a$ s.t. $\Pr_{h \leftarrow H^{\ell}_{1/2-2\epsilon}}[h \in A_{\alpha'}] \geq \frac{1}{2} \cdot 2^{-a} = 2^{-(a+1)}$.

Let $R = H^{\ell}_{1/2-2\epsilon}$ and $Z = (R | R \in A_{\alpha'})$, following [27, 10] we would like to argue that (in some sense to be explained below) for every $x \in \{0,1\}^k$, $\mathrm{Red}^{(\cdot)}(x, \alpha')$ does not distinguish between the oracle $f^{\oplus t} \oplus Z$ (in which bits can be correlated in complicated ways) and the oracle $f^{\oplus t} \oplus R$ (in which bits are ℓ-wise independent). Note that for every $x \in \{0,1\}^k$, and $h : \{0,1\}^n \to \{0,1\}$, we can think of $\mathrm{Red}^{f^{\oplus t} \oplus h}(x, \alpha')$ as a decision tree (that depends on x) that makes q queries to (the truth table of) h.

With this intuition in mind, we will prove the following lemma (which generalizes a "fixed set lemma" proven in [10] for the special case where the random variables $R_1, \ldots, R_N$ are independent).

▶ **Lemma 20** (A more general fixed set lemma)**.** *Let N, a and q be integers, and let $R = (R_1, \ldots, R_N)$ be some distribution, let $A \subseteq \{0,1\}^N$ be an event such that $\Pr[R \in A] \geq 2^{-a}$, and let $Z = (R | R \in A)$. For every $\eta > 0$, there exists a set $B \subseteq [N]$ of size $b \leq O(a \cdot q/\eta)$, and $v \in \{0,1\}^B$ in the support of Z_B, such that for $R' = (R | R_B = v)$ and $Z' = (Z | Z_B = v) = (R | R_B = v, R \in A)$, and every q-query decision tree P, $P(R')$ and $P(Z')$ are η-close.*[8]

Loosely speaking, the proof works by showing that if there exists a q-query decision tree that distinguishes R from Z, then by fixing the variables on some path of that tree, one obtains a distribution R' such that $\Pr[R' \in A] \geq \Pr[R \in A] \cdot (1 + \eta)$. We apply this argument iteratively (using R' as R) until there does not exist a q query decision tree that distinguishes

[8] Two distributions X, Y over the same domain S are η-close if for every $A \subset S$, $|\Pr[X \in A] - \Pr[Y \in A]| \leq \eta$.

R from Z. In each iteration, $\Pr[R \in A]$ increases by a factor of $1+\eta$, and as this probability cannot be larger than one, this process has to stop after $O(a/\eta)$ steps. By then, we have fixed no more than $O(qa/\eta)$ of the variables. The full proof of Lemma 20 appears in Section 4.[9]

We now continue with the proof of Lemma 11. We apply Lemma 20 on $R = H^{\ell}_{1/2-2\epsilon}$ and the event A'_{α}, using $\eta = \delta$, and let Z, R', Z', B, v and b be as in the lemma. It follows that for every $x \in \{0,1\}^k$, the random variables $\mathrm{Red}^{f^{\oplus t} \oplus R'}(x, \alpha')$ and $\mathrm{Red}^{f^{\oplus t} \oplus Z'}(x, \alpha')$ are δ-close. As this holds for every fixed $x \in \{0,1\}^k$, this also holds for an independently chosen $X \leftarrow U_k$, and we obtain that:

$$\Pr[\mathrm{Red}^{f^{\oplus t} \oplus R'}(X, \alpha') = f(X)] \geq \Pr[\mathrm{Red}^{f^{\oplus t} \oplus Z'}(X, \alpha') = f(X)] - \delta.$$

The support of Z' is contained in $A_{\alpha'}$, and so, for every h in the support of Z', $(f^{\oplus t} \oplus h)$ is useful (with the advice string α') and we get that:

$$\Pr_{X \leftarrow U_k}[\mathrm{Red}^{f^{\oplus t} \oplus Z'}(X, \alpha') = f(X)] \geq 1 - \delta. \tag{1}$$

Combining this with the previous inequality, gives that:

$$\Pr[\mathrm{Red}^{f^{\oplus t} \oplus R'}(X, \alpha') = f(X)] \geq 1 - 2\delta. \tag{2}$$

The advantage of (2) over (1) is that we have replaced Z' (in which the bits of $Z'([N] \setminus B)$ can be correlated in complicated ways) with R', where $R'([N] \setminus B)$ is $(\ell - b)$-wise independent. This will allow us to relate this oracle to $\mathsf{Noise}^q_{1/2-2\epsilon}$ and prove the zoom lemma.

2.5 Constructing the circuits for the zoom lemma

In order to construct the circuits required for the zoom lemma, we define the following oracle circuit.

▶ **Definition 21.** *We define an oracle circuit $E^{(\cdot)}(x)$ as follows: On input x and oracle h, $E^h(x)$ simulates* $\mathrm{Red}^{(\cdot)}(x, \alpha')$. *Whenever* Red *makes a query y to its oracle, R acts as follows: if $y \notin B$, then R makes the query y to h, and returns $f^{\oplus t}(y) \oplus h(y)$ to* Red. *If $y \in B$, then R returns $f^{\oplus t}(y) \oplus v(y)$ to* Red. *The output of R is the output of* Red *at the end of this simulation.*

With this definition, it is possible to show that:

▶ **Lemma 22.** *By choosing the constant c_2 and the polynomial p_0 to be sufficiently large, we get that:*

- $\Pr[E^{H^q_{1/2-2\epsilon}}(X) = f(X)] \geq 1 - 2\delta$.
- $\Pr[E^{H^q_{1/2}}(X) = f(X)] \leq \frac{1}{2} + \frac{1}{200}$.
- *For every $x \in \{0,1\}^k$, there exists a circuit $T_x : \{0,1\}^q \to \{0,1\}$ of size* $\mathrm{poly}(r)$ *and depth $O(d)$ over the gates G, such that for every $0 \leq p \leq 1$, $T_x(\mathsf{Noise}^q_p) = E^{H^q_p}(x)$.*

We note that this lemma immediately implies Lemma 11. The proof of Lemma 22 appears in Section 3, and is similar in spirit to earlier work [27, 12, 10]. It is in fact significantly simpler, as in this paper, we have the additional advantage that $f^{\oplus t}$ has circuits of size $\mathrm{poly}(r)$ and constant depth.

---

[9] The proof of the fixed set lemma given in [10] also uses an iterative argument: It shows that the existence of a q-query decision tree gives rise to a new distribution Z where the entropy of Z is increased. It is then argued that the iterative process has to stop before (as the entropy of Z is upper bounded by N). This limits the earlier proofs to distributions R where $Z = (R|R \in A)$ has very high entropy, which isn't the case for our choice of oracle.

Organization of the paper

We prove Lemma 22 in Section 3. In Section 4 we prove the more general fixed set lemma (that is Lemma 20). In Appendix A we state and survey the results of [27] showing that the zoom lemma implies the main theorem. In Appendix B we explain how to extend the argument to sufficiently explicit linear codes.

3 Proof of Lemma 22

In this section we prove Lemma 22. We start by proving the first item. Note that $b = O(qa/\delta)$ is bounded by some polynomial in r. We are allowed to choose the polynomial p_0 to be sufficiently large so that $\ell = p_0(r)$ satisfies $(\ell - b) \geq q$. This gives that the $N - b$ coordinates of $R'([N] \setminus B)$ are $(\ell - b)$-wise independent (because R' was obtained by fixing b indices of R which is ℓ-wise independent). The fact that $R'([N] \setminus B)$ are q-wise independent, and that E answers queries in B using v, gives that for every $x \in \{0,1\}^k$, the q queries made by $E^{H^q_{1/2-2\epsilon}}(x)$ are distributed exactly like the queries of $\mathrm{Red}^{f^{\oplus t} \oplus R'}(x, \alpha')$, meaning that:

$$\Pr[E^{H^q_{1/2-2\epsilon}}(x) = f(x)] = \Pr[\mathrm{Red}^{f^{\oplus t} \oplus R'}(x, \alpha') = f(x)].$$

This immediately means that for an independent $X \leftarrow U_k$:

$$\Pr[E^{H^q_{1/2-2\epsilon}}(X) = f(X)] = \Pr[\mathrm{Red}^{f^{\oplus t} \oplus R'}(X, \alpha') = f(X)].$$

We have already seen in (2) that:

$$\Pr[\mathrm{Red}^{f^{\oplus t} \oplus R'}(X, \alpha') = f(X)] \geq 1 - 2\delta,$$

and this gives the first item.

For the second item, we note that if $E^{H^q_{1/2}}$ makes a query $y \notin B$, then it obtains a uniform coin, and the coins obtained on different queries are independent. Recall that on queries $y \in B$, E answers the queries without consulting the oracle. This means that we can simulate $E^{H^q_{1/2}}(x)$ by a randomized circuit $\bar{C}$ that on input x, simulates E and answers queries $y \notin B$ by random coins. It follows that for every $x \in \{0,1\}^k$:

$$\Pr[E^{H^q_{1/2}}(x) = f(x)] = \Pr[\bar{C}(x) = f(x)].$$

This immediately means that for an independent $X \leftarrow U_k$:

$$\Pr[E^{H^q_{1/2}}(X) = f(X)] = \Pr[\bar{C}(X) = f(X)].$$

There exists some fixing for the random coins of $\bar{C}$ such that the obtained (deterministic) circuit C satisfies $Pr[C(X) = f(X)] \geq \Pr[\bar{C}(X) = f(X)]$. The circuit C is hardwired with this choice of random coins, and with α', B, v, and $f^{\oplus t}(B)$. (A crucial observations is that C does not need to compute $f^{\oplus t}$ for $y \notin B$). Overall, this is a circuit of size r^c for some constant c, and by choosing the constant c_2 from to be a larger constant, and using Lemma 13, we have that:

$$\Pr[E^{H^q_{1/2}}(X) = f(X)] \leq \Pr[C(X) = f(X)] \leq \frac{1}{2} + \frac{1}{200}.$$

This proves the second item.

For the third item, we note (once again) that for every p, and for every $x \in \{0,1\}^k$, the distribution of the q answers that $E^{H^q_p}$ obtains from its oracle is distributed like Noise^q_p. This means that for every x, we can construct a circuit T_x that on input Noise^q_p simulates $E^{H^q_p}(x)$,

using its i'th input to answer the i'th query of E. The circuit T_x is hardwired with α', B and v. Unlike the circuit C from the second item, T_x needs to compute $f^{\oplus t}$ on each of the q queries. This can be done using Corollary 14. Overall, T_x is a circuit of size $\text{poly}(r)$ (this time the polynomial is larger than r^{c_2}) and depth $O(d)$ (because on every oracle call of Red, T_x may have to compute $f^{\oplus t}$ (which takes constant depth according to Corollary 14).[10]

4 The fixed set lemma for ℓ-wise independence

In this section we prove Lemma 20. The proof will iteratively applying the following lemma.

▶ **Lemma 23.** *Let $R = (R_1, \ldots, R_N)$ be a distribution, let $A \subseteq \{0,1\}^N$ be an event, and let $Z = (R|R \in A)$. If there exists a q-query decision tree P such that $|\Pr[P(R) = 1] - \Pr[P(Z) = 1]| > \eta$ then there exists $Q \subset [N]$ of size q and $v \in \{0,1\}^Q$ in the support of Z_Q, such that*

$$\Pr[R \in A | R(Q) = v] > (1 + \eta) \cdot \Pr[R \in A].$$

Proof. Let P be a q-query decision tree, and assume w.l.o.g. (by complementing P if necessary) that $\Pr[P(Z) = 1] - \Pr[P(R) = 1] > \eta$. A path in the decision tree corresponds to a subset $Q \subset [N]$ of the q variables queried on the path, and a string $v \in \{0,1\}^q$ of the answers. For every such path, let $path_{Q,v} : \{0,1\}^N \to \{0,1\}$ be the function that evaluates to 1 on input $r = (r_1, \ldots, r_N)$ if $r(Q) = v$ (meaning that the tree P takes the path (Q, v) on input r). Let S be the set of all pairs (Q, v) corresponding to paths of P that answer 1. The path taken by a decision tree is unique, and therefore, for any distribution R on $\{0,1\}^N$, we have that:

$$\Pr[P(R) = 1] = \sum_{(Q,v) \in S} \Pr[path_{Q,v}(R) = 1].$$

▷ Claim 24. There exists a path $(Q, v) \in S$ such that:

$$\Pr[path_{Q,v}(Z) = 1] > (1 + \eta) \cdot \Pr[path_{Q,v}(R) = 1].$$

Proof of claim. This is because otherwise:

$$\begin{aligned}
\Pr[P(Z) = 1] &= \sum_{(Q,v) \in S} \Pr[path_{Q,v}(Z) = 1] \\
&\leq (1 + \eta) \cdot \sum_{(Q,v) \in S} \Pr[path_{Q,v}(R) = 1] \\
&= (1 + \eta) \cdot \Pr[P(R) = 1] \\
&\leq \Pr[P(R) = 1] + \eta.
\end{aligned}$$

◁

[10] Our proof of Lemma 22 relies on the fact that $f^{\oplus t}$ has small constant depth circuits. This allows us to simplify the argument used by some of the previous work [27, 12, 10] which wasn't allowed to assume that the target function $f' = f^{\oplus t}$ can be computed by a small constant depth circuit. The proofs in [27, 12, 10] need to resort to different arguments (and this creates additional difficulties if Red makes adaptive calls to its oracle, meaning that the queries that $\text{Red}^{(\cdot)}(x, \alpha')$ makes are not a function of x and α', and may also depend on previous answers). However, using a clever hybrid argument of [12] and additional ideas explained in [10], it is possible to conclude that T_x has depth $O(d)$ *without* relying on the fact that f' is computable by constant depth circuits. This argument allows choosing f' where $f' = \text{Con}^f$ for an oracle circuit Con that has size $\text{poly}(r)$, but does not necessarily has constant depth, and this gives the aforementioned extension of Theorem 9 to this setup, which now holds for every class $\mathcal{D}$ that contains circuits of size $\text{poly}(r)$.

In particular, the event $\{R(Q) = v\}$ occurs with positive probability, and it follows that:

$$\begin{aligned}\Pr[R \in A | R(Q) = v] &= \frac{\Pr[R \in A \cap R(Q) = v]}{\Pr[R(Q) = v]} \\ &= \frac{\Pr[R \in A] \cdot \Pr[R(Q) = v | R \in A]}{\Pr[path_{Q,v}(R) = 1]} \\ &= \frac{\Pr[R \in A] \cdot \Pr[Z(Q) = v]}{\Pr[path_{Q,v}(R) = 1]} \\ &= \frac{\Pr[R \in A] \cdot \Pr[path_{Q,v}(Z) = 1]}{\Pr[path_{Q,v}(R) = 1]} \\ &> (1+\eta) \cdot \Pr[R \in A].\end{aligned}$$

◀

We are now ready to prove Lemma 20.

Proof of Lemma 20. We consider the following iterative process: At step i, we have:

- A distribution $R^{(i)}$ over $\{0,1\}^N$.
- A set $B^{(i)} \subseteq [N]$.
- $v^{(i)} \in \{0,1\}^{B^{(i)}}$.

We will assume that the following invariant is satisfied:

- $B^{(i)}$ is of size $i \cdot q$.
- $\Pr[R^{(i)} \in A] \geq 2^{-a} \cdot (1+\eta)^i$.
- $R^{(i)}(B^{(i)}) = v^{(i)}$ (with probability one).

Note that the assumption in the lemma fulfills this invariant for $i = 0$ with $R^{(0)} = R.$ and $B^{(0)} = \emptyset$.

At step i, we define $\bar{R} = R^{(i)}([N] \setminus B^{(i)})$. As $R^{(i)}$ is fixed on $B^{(i)}$, we can think of A as an event that only observes the indices in $[N] \setminus B^{(i)}$. More formally, there is an event $\bar{A} \subseteq \{0,1\}^{[N] \setminus B^{(i)}}$ such that $R^{(i)} \in A$ iff $\bar{R} \in \bar{A}$, and

$$\Pr[\bar{R} \in \bar{A}] = \Pr[R^{(i)} \in A] \geq 2^{-a} \cdot (1+\eta)^i.$$

Let $\bar{Z} = (\bar{R} | \bar{R} \in \bar{A})$. If the conclusion of Lemma 20 does not hold with respect to $B^{(i)}, v^{(i)}$, then there exists a q-query decision tree P that distinguishes $R^{(i)}$ from $(R^{(i)} | R^{(i)} \in A)$, with advantage η, and as the two distributions agree on the queries in $B^{(i)}$, we conclude that P distinguishes $\bar{R}$ from $\bar{Z}$ with the same advanatge. We apply Lemma 23 on $\bar{R}$ and $\bar{A}$, and conclude that there exists $Q \subseteq [N] \setminus B^{(i)}$ and $v \in \{0,1\}^Q$ such that

$$\Pr[\bar{R} \in \bar{A} | \bar{R}(Q) = v] > (1+\eta) \cdot \Pr[\bar{R} \in \bar{A}] \geq 2^{-a} \cdot (1+\eta)^{i+1}.$$

We set:

- $B^{(i+1)} = B^{(i)} \cup Q$.
- $v^{(i+1)}$ to be the "concatenation of $v^{(i)}$ and v". More precisely, for $y \in B^{(i)}$, $v_y^{(i+1)} = v_y^{(i)}$ and for $y \in Q$, $v_y^{(i+1)} = v_y$.
- $R^{(i+1)} = (R^{(i)} | R^{(i)}(Q) = v)$. (Note that by definition $B^{(i)} \cap Q = \emptyset$).

We now observe that the invariant is maintained in step $i+1$. Specifically:

- $|B^{(i+1)}| = |B^{(i)}| + q = i \cdot q + q = (i+1) \cdot q$.
- $\Pr[R^{(i+1)} \in A] = \Pr[R^{(i)} \in A | R^{(i)}(Q) = v] = Pr[\bar{R} \in \bar{A} | \bar{R}(Q) = v] \geq 2^{-a} \cdot (1+\eta)^{i+1}$.
- By definition, $R^{(i+1)}(B^{(i+1)}) = v^{(i+1)}$ with probability one.

Therefore, if this process fails to deliver the lemma after i steps, then the invariant is maintained at the end of step i, and in particular, $\Pr[R^{(i)} \in A] \geq 2^{-a} \cdot (1+\eta)^i$. However, this is impossible for $i > \frac{a}{\log(1+\eta)} = \Theta(a/\eta)$, and so, this process has to deliver the lemma within this number of steps. We obtain that the lemma follows with $b = |B| \leq O(\frac{qa}{\eta})$. ◀

5 Conclusion and open problems

Class reductions are known to bypass some limitations on black-box reductions (as explained in Section 1.2). This work demonstrates that it is sometimes possible to extend limitations on black-box reductions to class reductions. Studying the power of class reductions may promote our understanding of how to bypass limitations on black-box reductions. We now mention some more specific open problems:

- Unlike the results of [10], our results do not hold for any construction of target functions f' from f. Is it possible to extend our results to this general setting?
- In Theorem 9, the class $\mathcal{D}$ contains circuits that are polynomially larger than the size of the the reduction. Is it possible to extend our limitations on class reductions with respect to a classes $\mathcal{D}$ of circuits smaller than the circuit size of the reduction?
- Yao's XOR lemma states that *for every* function f, if f is somewhat hard, then $f^{\oplus t}$ is very hard. It makes sense to focus on some specific choice for a somewhat hard function f and prove and improved result for this specific function. If we prove such an assertion by reduction, we can allow the reduction to be tailored to the specific function f, and do no need to require that the reduction performs on any function f, but only on the chosen one. This type of reductions was termed "function specific" by Artemenko and Shaltiel [3], who proved limitations on nonuniform black-box functions specific reductions. It is interesting to understand whether function specific class reductions can circumvent the limitations proven in this paper. We remark that our proof technique indeed relies on the fact that the reduction is not function specific, and must work for *any* function f. This allows us to choose f with specific properties that are useful for our argument.

References

1 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on np-hardness. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 701–710, 2006. doi:10.1145/1132516.1132614.

2 Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. *Computational Complexity*, 25(2):349–418, 2016. doi:10.1007/s00037-016-0128-9.

3 Sergei Artemenko and Ronen Shaltiel. Lower bounds on the query complexity of non-uniform and adaptive reductions showing hardness amplification. *Computational Complexity*, 23(1):43–83, 2014. doi:10.1007/s00037-012-0056-2.

4 Albert Atserias. Distinguishing SAT from polynomial-size circuits, through black-box queries. In *21st Annual IEEE Conference on Computational Complexity*, pages 88–95, 2006. doi:10.1109/CCC.2006.17.

5 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.

6 Bill Fefferman, Ronen Shaltiel, Christopher Umans, and Emanuele Viola. On beating the hybrid argument. *Theory of Computing*, 9:809–843, 2013.

7 Joan Feigenbaum and Lance Fortnow. Random-self-reducibility of complete sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.

8 Oded Goldreich and Hugo Krawczyk. On the composition of zero-knowledge proof systems. *SIAM J. Comput.*, 25(1):169–192, 1996. doi:10.1137/S0097539791220688.

9 Oded Goldreich, Noam Nisan, and Avi Wigderson. On yao's xor-lemma. In Oded Goldreich, editor, *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, volume 6650 of *Lecture Notes in Computer Science*, pages 273–301. Springer, 2011. doi:10.1007/978-3-642-22670-0_23.

10 Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *59th IEEE Annual Symposium on Foundations of Computer Science*, pages 956–966, 2018. doi:10.1109/FOCS.2018.00094.

11 Dan Gutfreund. Worst-case vs. algorithmic average-case complexity in the polynomial-time hierarchy. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 9th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX and 10th International Workshop on Randomization and Computation, RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 386–397, 2006. doi:10.1007/11830924_36.

12 Dan Gutfreund and Guy N. Rothblum. The complexity of local list decoding. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX, and 12th International Workshop, RANDOM*, volume 5171 of *Lecture Notes in Computer Science*, pages 455–468, 2008. doi:10.1007/978-3-540-85363-3_36.

13 Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. If NP languages are hard on the worst-case, then it is easy to find their hard instances. *Computational Complexity*, 16(4):412–441, 2007. doi:10.1007/s00037-007-0235-8.

14 Dan Gutfreund and Amnon Ta-Shma. Worst-case to average-case reductions revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 10th International Workshop, APPROX, and 11th International Workshop, RANDOM*, volume 4627 of *Lecture Notes in Computer Science*, pages 569–583, 2007. doi:10.1007/978-3-540-74208-1_41.

15 Dan Gutfreund and Salil P. Vadhan. Limitations of hardness vs. randomness under uniform reductions. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques, 11th International Workshop, APPROX, and 12th International Workshop, RANDOM*, volume 5171 of *Lecture Notes in Computer Science*, pages 469–482, 2008. doi:10.1007/978-3-540-85363-3_37.

16 Dan Gutfreund and Emanuele Viola. Fooling parity tests with parity gates. In *Approximation, Randomization, and Combinatorial Optimization, Algorithms and Techniques, 7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX, and 8th International Workshop on Randomization and Computation, RANDOM*, volume 3122 of *Lecture Notes in Computer Science*, pages 381–392, 2004. doi:10.1007/978-3-540-27821-4_34.

17 Shuichi Hirahara. Non-black-box worst-case to average-case reductions within NP. In *59th IEEE Annual Symposium on Foundations of Computer Science*, pages 247–258, 2018. doi:10.1109/FOCS.2018.00032.

18 Russell Impagliazzo. Hard-core distributions for somewhat hard problems. In *36th Annual Symposium on Foundations of Computer Science*, pages 538–545, 1995. doi:10.1109/SFCS.1995.492584.

19 Russell Impagliazzo and Steven Rudich. Limits on the provable consequences of one-way permutations. In *Proceedings of the 21st Annual ACM Symposium on Theory of Computing*, pages 44–61, 1989. doi:10.1145/73007.73012.

20 Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing*, pages 220–229, 1997. doi:10.1145/258533.258590.

21 Adam R. Klivans and Rocco A. Servedio. Boosting and hard-core set construction. *Machine Learning*, 51(3):217–238, 2003. doi:10.1023/A:1022949332276.

22 Nutan Limaye, Karteek Sreenivasaiah, Srikanth Srinivasan, Utkarsh Tripathi, and S. Venkitesh. A fixed-depth size-hierarchy theorem for $\mathrm{ac}^0[\oplus]$ via the coin problem. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing*, pages 442–453, 2019. doi:10.1145/3313276.3316339.

23 Chi-Jen Lu, Shi-Chun Tsai, and Hsin-Lung Wu. On the complexity of hardness amplification. *IEEE Trans. Information Theory*, 54(10):4575–4586, 2008. doi:10.1109/TIT.2008.928988.

24 Igor Carboni Oliveira, Rahul Santhanam, and Srikanth Srinivasan. Parity helps to compute majority. In *34th Computational Complexity Conference*, volume 137, pages 23:1–23:17, 2019. doi:10.4230/LIPIcs.CCC.2019.23.

25 Alexander Razborov. Lower bounds on the dimension of schemes of bounded depth in a complete basis containing the logical addition function. *Akademiya Nauk SSSR. Matematicheskie Zametki*, 41(4):598–607, 1987. English translation in Mathematical Notes of the Academy of Sci. of the USSR, 41(4):333-338, 1987.

26 Omer Reingold, Luca Trevisan, and Salil P. Vadhan. Notions of reducibility between cryptographic primitives. In *Theory of Cryptography, First Theory of Cryptography Conference, TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 1–20, 2004. doi:10.1007/978-3-540-24638-1_1.

27 Ronen Shaltiel and Emanuele Viola. Hardness amplification proofs require majority. *SIAM J. Comput.*, 39(7):3122–3154, 2010. doi:10.1137/080735096.

28 Roman Smolensky. Algebraic methods in the theory of lower bounds for boolean circuit complexity. In *Proceedings of the 19th Annual ACM Symposium on Theory of Computing*, pages 77–82, 1987. doi:10.1145/28395.28404.

29 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom generators without the XOR lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

30 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and average-case complexity via uniform reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

31 Emanuele Viola. Hardness vs. randomness within alternating time. In *18th Annual IEEE Conference on Computational Complexity*, page 53, 2003. doi:10.1109/CCC.2003.1214410.

32 Emanuele Viola. *The Complexity of Hardness Amplification and Derandomization*. PhD thesis, Harvard University, 2006.

A Showing that the main theorem follows from the zoom lemma

In this section we show that Theorem 9 follows from Lemma 11. This follows by the earlier work of Shaltiel and Viola [27] which we now explain.

A.1 Consequences distinguishing noise $\frac{1}{2}$ from $(\frac{1}{2} - 2\epsilon)$

The next lemma shows that distinguishing between $\mathsf{Noise}^q_{1/2-2\epsilon}$ and $\mathsf{Noise}^q_{1/2}$ requires many queries.

▶ **Lemma 25** ([32, 27]). *For every $\epsilon, \delta > 0$, such that $\delta < 0.4$, if $T : \{0,1\}^q \to \{0,1\}$ satisfies:*

- $\Pr[T(\mathsf{Noise}^q_{1/2-2\epsilon}) = 1] \geq 1 - \delta$.
- $\Pr[T(\mathsf{Noise}^q_{1/2}) = 1] \leq 0.51$.

Then, $q = \Omega(\frac{\log \frac{1}{\delta}}{\epsilon})$.

The next lemma essentilly shows that distinguishing between $\mathsf{Noise}^q_{1/2-2\epsilon}$ and $\mathsf{Noise}^q_{1/2}$ requires majority on $\Omega(1/\epsilon)$ bits. A technicality is that for this conclusion it is not sufficient to distinguish $\mathsf{Noise}^q_{1/2-2\epsilon}$ from $\mathsf{Noise}^q_{1/2}$, and one needs circuits that distinguish $\mathsf{Noise}^q_{1/2-\frac{2}{j}}$ from $\mathsf{Noise}^q_{1/2}$ for every integer j between 1 and $\log(1/\epsilon)$. This complication is in some sense necessary (see discussion in [27]).

▶ **Lemma 26** ([32, 27]). *For every $\epsilon, \delta > 0$, such that $\delta < 0.4$, and $\frac{1}{\epsilon}$ is an integer. If $T_1, \ldots, T_{\frac{1}{\epsilon}}$ are circuits over q bits, with size $s \geq q$ and depth d (over some set of gates G that includes the standard set $\{\text{AND}, \text{OR}, \text{NOT}\}$ with unbounded fan-in) and for every $j \in [\frac{1}{\epsilon}]$, we have that:*

- $\Pr[T_j(\mathit{Noise}^q_{1/2-\frac{2}{j}}) = 1] \geq 1 - \delta$.
- $\Pr[T_j(\mathit{Noise}^q_{1/2}) = 1] \leq 0.51$.

Then, there exists a circuit A that computes the majority function over $\Omega(\frac{1}{\epsilon})$ bits, and A has size $s \cdot \mathrm{poly}(\frac{1}{\epsilon})$ and depth $d + O(1)$ over the same set of gates G.

A.2 Finishing up

We now have have all the tools to prove that Theorem 9 follows from Lemma 11. We will assume w.l.o.g. that $\frac{1}{\epsilon}$ is an integer. We first observe that a $(\frac{1}{2} + \epsilon) \to (1 - \delta)$ class $\mathcal{D}$ reduction for Yao's XOR lemma, is in particular a $(\frac{1}{2} + \frac{1}{j}) \to (1 - \delta)$ class $\mathcal{D}$ reduction for Yao's XOR lemma, for every $j \in [\frac{1}{\epsilon}]$.

This means that for every $j \in [\frac{1}{\epsilon}]$, we can apply Lemma 11 choosing the parameter ϵ to be $\epsilon = \frac{1}{j}$, and for each such j and $x \in \{0,1\}^k$ we obtain a circuit T^j_x over q bits with size $\mathrm{poly}(r)$ and depth $O(d)$ over the set of gates G such that:

- $\Pr_{X \leftarrow U_k}[T^j_X(\mathsf{Noise}^q_{1/2-\frac{2}{j}}) = 1] \geq 1 - 2\delta$.
- $\Pr_{X \leftarrow U_k}[T^j_X(\mathsf{Noise}^q_{1/2}) = 1] \leq \frac{1}{2} + \frac{1}{200}$.

Applying Markov's inequality to the second item of the lemma, we obtain that there exists a constant $\beta > 0$ such that for every j, for a β fraction of $x \in \{0,1\}^k$,

$$\Pr[T^j_x(\mathsf{Noise}^q_{1/2}) = 1] \leq 0.51.$$

Applying Markov's inequality to the first item of the lemma, we obtain that for every j, for a $1 - \beta/2$ fraction of $x \in \{0,1\}^k$,

$$\Pr[T^j_x(\mathsf{Noise}^q_{1/2-\frac{2}{j}}) = 1] \leq 1 - \frac{4 \cdot \delta}{\beta} \leq 1 - \frac{4 \cdot \delta_0}{\beta}.$$

Together, this gives that for every j, there exists $x \in \{0,1\}^k$ that satisfies both inequalities. Theorem 9 now follows directly from Lemma 25 and Lemma 26, by choosing the constant $\delta_0 > 0$ to be sufficiently small.[11]

B Extending the argument to sufficiently explicit linear codes

We now prove the extension of our results to sufficiently explicit linear codes which is stated in Section 1.3. More specifically, we will prove a version of Theorem 9 that assumes that $\delta = 2^{-2k}$ and replace replace $f^{\oplus t}$ by a function $f' : \{0,1\}^n \to \{0,1\}$ defined by:

$$f'(y) = \sum_{x \in \{0,1\}^k} f(x) \cdot g(x,y),$$

where the sum is taken in the field $\mathbb{F}_2$, and $g : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}$ can be computed by circuits of size $\mathrm{poly}(r)$ and depth d over the set G of gates.

This argument is based on a trick by Viola [32] that we can incorporate into our framework. We will modify the proof of Lemma 11 so that it holds in this setting, the modified version of Theorem 9 will follow from Lemma 11 just as before.

We start by replacing the function f of Lemma 13 with a sightly different function:

[11] The argument above is wasteful, and leads to a rather small constant $\delta_0 > 0$. We remark that with a more careful argument, we could have chosen δ_0 to be any constant smaller than $\frac{1}{2}$, and even allow it to approach $\frac{1}{2}$. More specifically, a more careful analysis can allow $\delta_0 = \frac{1}{2} - O(\log(1/\epsilon))$.

▶ **Lemma 27.** *There exist constants c_1 such that for every constant c_2, there exists a function $f : \{0,1\}^k \to \{0,1\}$ such that there exists $m \leq c_1 \cdot c_2 \cdot \log r$ such that:*

- *For every circuit $B : \{0,1\}^k \to \{0,1\}$ of size r^{c_2}, $\Pr_{X \leftarrow U_m}[B(X \circ 0^{k-m}) = f(X \circ 0^{k-m})] \leq \frac{1}{2} + \frac{1}{200}$.*
- *f can be computed by a DNF of size $r^{c_1 \cdot c_2}$.*

Proof. We repeat the proof of Lemma 13, but this time we take $f(x)$ to be the following function: Let x' denote the first m bits of x and x'' denote the remaining $k-m$ bits. We define $f(x)$ to be $g(x')$ if $x'' = 0^{k-m}$ and zero otherwise. ◀

On a random $X \leftarrow U_k$, $\Pr[f(X) = 0] \geq 1 - 2^{k-m} = 1 - 2^{-\Omega(k)}$. Therefore, it is very easy to compute f with success probability $1 - 2^{-\Omega(k)}$ by simply answering zero. However, by Lemma 27 it is hard for circuits of size r^{c_2} to compute f with success probability 1, or equivalently success probability $1 - \delta$ for $\delta = 2^{-2k}$. This is why this approach can only succeed for very small δ.

With this choice, we can get a corollary that is analogous to Corollary 14.

▶ **Corollary 28.** *The function f' can be computed by circuits of size* $\mathrm{poly}(r)$ *and constant depth over the gates* $\{$AND,OR,NOT,PARITY$\}$ *of unbounded fan-in.*

Proof. The function f' is defined by:

$$f'(y) = \sum_{x \in \{0,1\}^k} f(x) \cdot g(x,y).$$

The sum ranges over 2^k choices of x. However, for our function f, except for $\mathrm{poly}(r)$ of these x (the ones for which the second part of x is $k-m$ zeros) all the remaining x have $f(x) = 0$. This, together with the requirement on g, gives the required result. ◀

The proof proceeds as in Section 2, with the following modifications:

- For $\delta = 2^{-2k} < 2^{-k}$, Red is a $(\frac{1}{2} + \epsilon) \to 1$ reduction. This means in particular that if D is useful, then there exists $\alpha \in \{0,1\}^a$ such that $\Pr_{X \leftarrow U_m}[\mathrm{Red}^D(X \circ 0^{k-m}, \alpha) = f(X \circ 0^{k-m})] = 1$.
- We set $\delta' = \frac{1}{r}$ and will replace some occurrences of δ in the earlier argument by δ'. This is done because the choice of $\delta = 2^{-2k}$ does not satisfy the requirement that $\frac{1}{\delta} \leq r$ made in Theorem 9. Specifically, the requirement that $\frac{1}{\delta} \leq r$ was used to argue that when we apply Lemma 20 with $\eta = \delta$, the size of the set B (which is polynomial in $\frac{1}{\eta}$) is polynomial in r. In order to obtain a set B of size $\mathrm{poly}(r)$ we will now choose $\eta = \delta'$.
- In Section 2.4 we argued that for an independent $X \leftarrow U_k$:

  $$\Pr[\mathrm{Red}^{f^{\oplus t} \oplus R'}(X, \alpha') = f(X)] \geq \Pr[\mathrm{Red}^{f^{\oplus t} \oplus Z'}(X, \alpha') = f(X)] - \delta.$$

  With our modifications we get that for an independent $X \leftarrow U_m$:

  $$\Pr[\mathrm{Red}^{f' \oplus R'}(X \circ 0^{k-m}, \alpha') = f(X \circ 0^{k-m})] \geq \Pr[\mathrm{Red}^{f^{\oplus t} \oplus Z'}(X, \alpha') = f(X)] - \delta'.$$

- This allows us to continue the argument, replacing occurrences of $X \leftarrow U_k$ by $X \circ 0^{k-m}$ for $X \leftarrow U_m$, and occurrences of δ by δ'.
- When we finish the proof and obtain a lower bound of $q = \Omega(\frac{\log(1/\delta')}{\epsilon^2}) = \Omega(\frac{\log r}{\epsilon^2})$ as required. The result on majority is not affected by replacing δ by δ'.

Balanced Allocation on Dynamic Hypergraphs

Catherine Greenhill
UNSW Sydney, Australia
c.greenhill@unsw.edu.au

Bernard Mans
Macquarie University, Sydney, Australia
bernard.mans@mq.edu.au

Ali Pourmiri
Macquarie University, Sydney, Australia
ali.pourmiri@mq.edu.au

Abstract

The balls-into-bins model randomly allocates n sequential balls into n bins, as follows: each ball selects a set D of $d \geqslant 2$ bins, independently and uniformly at random, then the ball is allocated to a least-loaded bin from D (ties broken randomly). The *maximum load* is the maximum number of balls in any bin. In 1999, Azar et al. showed that, provided ties are broken randomly, after n balls have been placed the *maximum load*, is $\log_d \log n + \mathcal{O}(1)$, with high probability. We consider this popular paradigm in a dynamic environment where the bins are structured as a *dynamic hypergraph*. A dynamic hypergraph is a sequence of hypergraphs, say $\mathcal{H}^{(t)}$, arriving over discrete times $t = 1, 2, \ldots$, such that the vertex set of $\mathcal{H}^{(t)}$'s is the set of n bins, but (hyper)edges may change over time. In our model, the t-th ball chooses an edge from $\mathcal{H}^{(t)}$ uniformly at random, and then chooses a set D of $d \geqslant 2$ random bins from the selected edge. The ball is allocated to a least-loaded bin from D, with ties broken randomly. We quantify the dynamicity of the model by introducing the notion of *pair visibility*, which measures the number of rounds in which a pair of bins appears within a (hyper)edge. We prove that if, for some $\varepsilon > 0$, a dynamic hypergraph has pair visibility at most $n^{1-\varepsilon}$, and some mild additional conditions hold, then with high probability the process has maximum load $\mathcal{O}(\log_d \log n)$. Our proof is based on a variation of the witness tree technique, which is of independent interest. The model can also be seen as an adversarial model where an adversary decides the structure of the possible sets of d bins available to each ball.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases balls-into-bins, balanced allocation, power of two choices, witness tree technique

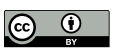
Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.11

Category RANDOM

Funding The first author is supported by the Australian Research Council Discovery Project DP190100977. The second and third authors are supported by the Australian Research Council Discovery Project DP170102794.

1 Introduction

The standard balls-into-bins model is a process that randomly allocates m sequential balls into n bins, where each ball chooses a set D of d bins, independently and uniformly at random, then the ball is allocated to a least-loaded bin from D (with ties broken randomly). When $m = n$ and $d = 1$, it is well known that at the end of process the maximum number of balls



© Catherine Greenhill, Bernard Mans, and Ali Pourmiri;
licensed under Creative Commons License CC-BY



at any bin, the *maximum load*, is $(1 + o(1))\frac{\log n}{\log \log n}$, with high probability. Surprisingly, Azar et al. [2] showed that for this d-choice process with $d \geqslant 2$, provided ties are broken randomly, the maximum load exponentially decreases to $\log_d \log n + \mathcal{O}(1)$. This phenomenon is known as the *power of d choices*. The multiple-choice paradigm has been successfully applied in a wide range of problems from nearby server selection, and load-balanced file placement in the distributed hash table, to the performance analysis of dictionary data structures (e.g., see [21]). In the classical setting, all $\binom{n}{d}$ sets of d bins are available to each ball. However, in many realistic scenarios such as cache networks, peer-to-peer or cloud-based systems, the balls (requested files, jobs, items,..) have to be allocated to bins (servers, processors,...) that are close to them, in order to minimize the access latencies. On the other hand, the lack of perfect randomness stimulates the de-randomization of the d-choice process, which also requires the study of non-uniform distributions over choices (e.g. [1, 6, 7, 11]). Hence in many settings, allowing all possibilities for the set D of d bins is costly, and may not be practical. This motivates the investigation of the effect of distributions of the set D on the maximum load. In this regard, Kenthapadi and Panigrahy [13] proposed *balanced allocation* on graphs, where bins form the vertices of a Δ-regular graph and each ball chooses an edge of the graph uniformly at random. The ball is then placed in an endpoint of the selected edge with smaller load (ties are broken randomly). Kenthapadi and Panigrahy showed that the maximum load is $\Theta(\log \log n)$ if and only if $\Delta = n^{\Omega(1/\log \log n)}$. Here, one may see that the possibilities for the set D (the two chosen bins) is restricted to the set of $n\Delta/2$ edges of the graph. In the standard balls-into-bins model with $d = 2$, the underlying graph is a complete graph (all $\binom{n}{2}$ edges present). Following the study of balls-into-bins with related choices, Godfrey [12] utilized hypergraphs to model the structure of bins. In this model, each ball picks a random edge of a given hypergraph that contain $\Omega(\log n)$ bins and the hypergraph satisfies some mild conditions. Then, the ball is allocated to a least-loaded bin contained in the edge, with ties broken randomly. Godfrey showed that the maximum load is constant. Balanced allocation on graphs and hypergraphs has been further studied in [3, 4, 17, 18]. In the aforementioned works, either the underlying graph is fixed during the process or, in the hypergraph setting, the number d of choices satisfies $d = \Omega(\log n)$. However, in many real-world systems the structure may change over time, and probing the load of $\Omega(\log n)$ bins might be a costly task. Seeking a more realistic model, this paper studies the d-choice process in dynamic graphs and hypergraphs, where $2 \leqslant d = o(\log n)$.

Balanced allocation on dynamic hypergraphs can also be seen as an adversarial model, where the set D of potential choices is proposed by an adversary (environment) whose goal is to increase the maximum load. Here we want to understand the conditions under which the balanced allocation on dynamic (hyper)graphs still benefits from the effect of the power of d choices.

1.1 Our Results

We propose balanced allocation algorithms on different dynamic environments, namely dynamic graph and hypergraph models. In order to measure the dynamicity, we introduce the notion of *pair visibility*. For a pair $\{i, j\}$ of distinct vertices, the *visibility* of $\{i, j\}$, denoted by $\texttt{vis}(i, j)$, is the number of rounds $t \in \{1, \ldots, n\}$ such $\{i, j\}$ is contained in the edge chosen at round t. (A more formal definition is given below.) When ball i is placed into a bin, the *height* of ball i is the number of balls that were allocated to the bin before ball i. We say that event E_n holds with high probability (w.h.p.) if $\mathbf{Pr}[\mathsf{E}_n] \geqslant 1 - n^{-c}$ for every constant $c > 0$.

Balanced Allocation on Dynamic Hypergraphs

Write $[n] = \{1, \dots, n\}$ to be the set of n bins. A hypergraph $\mathcal{H} = ([n], \mathcal{E})$ is *s-uniform* if $|H| = s$ for every $H \in \mathcal{E}$. For every integer $n \geqslant 1$, let $s = s(n)$ be an integer such $2 \leqslant s \leqslant n$. A *dynamic s-uniform hypergraph*, denoted by $(\mathcal{H}^{(1)}, \mathcal{H}^{(2)}, \dots, \mathcal{H}^{(n)})$, is a sequence of s-uniform hypergraphs $\mathcal{H}^{(t)} = ([n], \mathcal{E}_t)$ with vertex set $[n]$. The edge sets $\mathcal{E}_t$ may change with t. A hypergraph is *regular* if every vertex is contained in the same number of edges.

In this paper, we are interested in the following properties which dynamic hypergraphs may satisfy. We refer to these properties as the *balancedness*, *visibility*, and *size* properties. The balancedness property is adapted from [3, 12].

Balancedness: Let H_t denote a randomly chosen edge from $\mathcal{E}_t$. If there exists a constant $\beta \geqslant 1$ such that $\mathbf{Pr}[i \in H_t] \leqslant \beta s/n$ for every $1 \leqslant t \leqslant n$ and each bin $i \in [n]$, then the dynamic hypergraph $(\mathcal{H}^{(1)}, \dots \mathcal{H}^{(n)})$ is β-balanced. A dynamic hypergraph is balanced if it is β-balanced for some constant $\beta \geqslant 1$. Every regular hypergraph is 1-balanced.

Visibility: For every pair of distinct vertices $\{i, j\} \subset [n]$, the visibility of $\{i, j\}$ is

$$\mathtt{vis}(i, j) = |\{t \in \{1, 2, \dots, n\} \mid \{i, j\} \subset H \in \mathcal{E}_t\}|.$$

If there exists a constant $\varepsilon \in (0, 1)$ such that $\mathtt{vis}(i, j) \leqslant sn^{1-\varepsilon}$ for all pairs $\{i, j\} \subseteq [n]$ of distinct bins then the dynamic hypergraph $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ is ε-visible. A dynamic hypergraph satisfies the visibility property if it is ε-visible for some constant $\varepsilon \in (0, 1)$.

Size: If $s = \Omega(\log n)$ and there exists a positive constant $c_0 \geqslant 1$ such that $|\mathcal{E}_t| \leqslant n^{c_0}$ for every $t \geqslant 1$, then the dynamic hypergraph $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ satisfies the c_0-size property. A dynamic hypergraph satisfies the size property if it satisfies the c_0-size property for some constant $c_0 \geqslant 1$.

▶ **Definition 1** (Balanced Allocation on Dynamic Hypergraphs). *Suppose that $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ is an s-uniform hypergraph and fix $d = d(n)$ with $2 \leqslant d = o(\log n)$ and $d \leqslant s$. The balanced allocation algorithm on $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ proceeds in rounds $(t = 1, 2, \dots, n)$, sequentially allocating n balls to n bins. In round t, the t-th ball chooses an edge H_t uniformly at random from $\mathcal{E}_t$, then it randomly chooses a set D_t of d bins from H_t (without repetition) and allocates itself to a least-loaded bin from D_t, with ties broken randomly.*

▶ **Theorem 2.** *Let $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ be a dynamic s-uniform hypergraph which satisfies the balancedness, ε-visibility and size properties. Fix $d = d(n)$ such that $2 \leqslant d = o(\log n)$. There exists $\Theta(n) \leqslant m \leqslant n$ such that after the balanced allocation process on $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ has allocated m balls, the maximum load is $\log_d \log n + \mathcal{O}(1/\varepsilon)$ with high probability. Moreover, for every fixed positive integer γ with $\gamma m \leqslant n$, after allocating γm balls the maximum load is at most $\gamma(\log_d \log n + \mathcal{O}(1/\varepsilon))$, w.h.p..*

▶ Remark 3. In our result we only consider the case where $d = o(\log n)$, because when $d = \Omega(\log n)$, a constant upper bound is obtained by [12]. The size property is mainly assumed for technical reasons. For instance, $|\mathcal{E}_t| \leqslant \mathrm{poly}(n)$ is not necessary. Roughly speaking, balanced allocation on a dynamic hypergraph with large $|\mathcal{E}_t|$ resembles the standard balls-into-bins process. So it might be possible that having more structural information about a dynamic hypergraph would enable us to extend our result to allow an arbitrary number of edges $|\mathcal{E}_t|$. Another possible extension of Theorem 2 would be to allow s to be a function of d.

Proof Technique. The proof of our main result (i.e., Theorem 2) is based on the witness tree technique, which has been applied in [1, 11, 13, 16, 20]. The standard 2-choice process simulates a random graph process, where any 2-element subset of bins chosen by a ball can be viewed as a random edge chosen from a complete graph. Kenthapadi and Panigrahi [13] replaced the underlying complete graph by a dense graph, say G, and each ball chooses a random edge from the graph. In this model, after allocating all n balls, the union of the chosen random edges builds a random subgraph of G. They showed that, with high probability, there does not exist a connected and random subgraph of size $\Omega(\log n)$ whose nodes (bins) each contain a constant number of balls. A rooted spanning tree contained in the subgraph is the witness structure. Now, if the maximum load is higher than a certain threshold, then a deterministic construction yields the witness graph, which is a rooted tree. Thus, the maximum load is bounded from above by the threshold. In this work, although we follow the same steps as [13], the witness structure is not as straightforward since the underlying structure is a dynamic hypergraph, $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$, and each ball chooses $d \geqslant 2$ bins. Here, the witness structure is a d-uniform hypergraph, say $H = (V, E)$, where (1) each node (bin) in V contains a constant number of balls, (2) $|V| = \Omega(\log n)$ and (3) H has an expansion property which means there is an ordering of the hyperedges of H so that with respect to this ordering, all but a constant number of hyperedges only shares one bin with the union of the preceding hyperedges. Applying the visibility condition we conclude that, with high probability, there does not exist a structure satisfying these three properties. Assuming a maximum load higher than a certain threshold, we recursively build the witness structure and the proof follows.

The following theorem presents a lower bound for the maximum load attained by the balanced allocation on some dynamic hypergraphs in terms of ε-visibility.

▶ **Theorem 4.** *Let $s = s(n) = n^{\varepsilon}$, where $\varepsilon \in (0, 1)$ is an arbitrary small real number. There exists a dynamic s-uniform hypergraph, say $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$, which satisfies the balancedness condition and (trivially) satisfies the ε-visibility condition. Let $2 \leqslant d \leqslant s$ be any integer which is constant. Suppose that the balanced allocation process on $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ has allocated n balls, then the maximum load is at least $\min\{\Omega(1/\varepsilon), \Omega(\log n/\log\log n)\}$ with high probability.*

Balanced Allocation on Dynamic Graphs

A dynamic graph is a special case of a dynamic hypergraph, where $s = s(n) = 2$ for all n. Write $(G^{(1)}, \ldots, G^{(n)})$ to denote a dynamic graph, where $G^{(t)} = ([n], E_t)$ for $t = 1, 2, \ldots, n$. Theorem 2 does not cover the case of graphs ($s = 2$), due to the size property. We will prove a result on balanced allocation for regular dynamic graphs.

▶ **Definition 5** (Balanced Allocation on Dynamic Graphs)**.** *Suppose that $(G^{(1)}, \ldots, G^{(n)})$ is a regular dynamic graph on vertex set $[n]$. The balanced allocation algorithm on $(G^{(1)}, \ldots, G^{(n)})$ proceeds in rounds $(t = 1, \ldots, n)$. In each round t, the t-th ball chooses an edge of $G^{(t)}$ uniformly at random, and the ball is then placed in one of the bins incident to the edge with a lesser load, with ties broken randomly.*

Say that the dynamic graph is *regular* if $G^{(t)}$ is Δ_t-regular for some positive integer Δ_t and all $t = 1, 2, \ldots, n$. For every pair of distinct bins $\{i, j\} \subset [n]$, we will assume that the visibility $\texttt{vis}(i, j)$ satisfies

$$\texttt{vis}(i, j) = |\{t \in \{1, 2, \ldots, n\} \mid \{i, j\} \in E_t\}| \leqslant 2n^{1-\varepsilon}$$

for some constant $\varepsilon \in (0, 1)$. This property is called ε-visibility.

▶ **Theorem 6.** *Let $(G^{(1)}, \dots, G^{(n)})$ be a regular dynamic graph which satisfies the ε-visibility condition, for some $\varepsilon \in (0,1)$. Suppose that the balanced allocation process on $(G^{(1)}, \dots, G^{(n)})$ has allocated n balls. Then the maximum load is at most $\log_2 \log n + \mathcal{O}(1/\varepsilon)$, with high probability.*

The proof, which can be found in Section 4, is again based on the witness tree technique. We remark that Theorem 6 can be extended to the case where the dynamic graph is *almost regular*, meaning that the ratio of the minimum and maximum degree of $G^{(t)}$ is bounded above by an absolute constant for $t = 1, \dots, n$.

Dynamic Graphs and Hypergraphs with Low Pair Visibility

In order to show the ubiquity of the visibility condition, we will describe some dynamic graphs with low pair visibility. One can easily construct a dynamic hypergraph from a dynamic graph by considering the r-neighborhood of each vertex of the t-th graph as a hyperedge in the t-th hypergraph, for $t = 1, \dots, n$.

- Dynamic Cycle. For $t = 1, \dots, n$ define the edge set

$$E_t = \{\{i,j\} \subset \{0, \dots, n-1\} \mid j = i + \lceil t/\sqrt{n} \rceil \pmod n \text{ or } i = j + \lceil t/\sqrt{n} \rceil \pmod n\},$$

where calculations are performed modulo n (that is, in the additive group $\mathbb{Z}_n$). In modular addition, for every pair $\{i,j\} \subset \{0, \dots, n-1\}$, the equation $i = j + k \pmod n$ has at most one solution $1 \leqslant k \leqslant \sqrt{n}$ and hence

$$\texttt{vis}(i,j) = |\{t \in \{1,2,\dots,n\} \mid \{i,j\} \in E_t\}| \leqslant \sqrt{n}.$$

Now $C^{(t)} = (\{0,1,\dots,n-1\}, E_t)$ is 2-regular, so it is either a Hamilton cycle or a union of two or more disjoint cycles (depending on whether t and n are coprime). By Theorem 6, the maximum load attained by the algorithm on $\{C^{(t)}, t = 1, \dots, n\}$ is at most $\log_2 \log n + \mathcal{O}(1)$. The analysis of the balanced allocation algorithm on Δ-regular graphs given by Kenthapadi and Panigrahy [13] showed that the balanced allocation process on arbitrary Δ-regular graphs has maximum load $\Theta(\log \log n)$ only when $\Delta = n^{\Omega(1/\log \log n)}$. By contrast, here each $C^{(t)}$ has degree at most 2, but the visibility condition keeps the maximum load as low as the standard two-choice process.

▶ Remark 7. By Theorem 6, w.h.p., the balanced allocation process on the dynamic cycle achieves the maximum load at most $\log_2 \log n + \mathcal{O}(1)$. Since $|E_t| = n$ for $t = 1, \dots, n$, each ball requires $\log_2 n$ random bits. However, in the standard power-of-two-choices process, each ball chooses two independent and random bins, which requires $2 \log n$ random bits. Therefore, the dynamic cycle can be used to reduce (by half) the number of random bits required in the standard two-choice process.

- Dynamic Modular Hypergraph. Suppose that n is a prime number and fix $s = s(n)$ such that $\log n \leqslant s \leqslant n^{1/5}$. (Here n is large enough so that this range is non-empty.) For $t = 1, \dots, n$, let $k_t = \lceil \sqrt{n} \rceil + \lceil \frac{t}{n^{3/4}} \rceil$ and for each $\alpha \in \mathbb{Z}_n$ define

$$H_t(\alpha) = \{\, \alpha + jk_t \pmod n \mid j = 0, 1, \dots, s-1 \,\}.$$

Then $H_t(\alpha)$ is a subset of $\mathbb{Z}_n$ of size s, as n is prime. Now for each $t = 1, \dots, n$ we define the dynamic s-uniform hypergraph $\mathcal{H}^{(t)} = (\mathbb{Z}_n, \mathcal{E}_t)$, where $\mathcal{E}_t = \{H_t(\alpha) \mid \alpha \in \mathbb{Z}_n\}$. Then $\mathcal{H}^{(t)}$ is s-regular, and hence 1-balanced, and it satisfies the 1-size property as $|\mathcal{E}_t| = n$.

Suppose that $\{\beta_1, \beta_2\} \subset H_t(\alpha)$ for some $\alpha \in \mathbb{Z}_n$, with $\beta_1 \neq \beta_2$. Then there exists $j_1, j_2 \in \{0, \ldots, s-1\}$ such that $\beta_1 = \alpha + j_1 k_t \pmod n$ and $\beta_2 = \alpha + j_2 k_t \pmod n$. Thus, $\beta_2 - \beta_1 = (j_2 - j_1) k_t \pmod n$. Note that j_1, j_2 must be distinct as β_1, β_2 are distinct. Next suppose that $k_{t_1} \neq k_{t_2}$ for some $t_1, t_2 \in \{1, \ldots, n\}$, and take any $j_1, j_2 \in \{1, \ldots, s-1\}$. By definition of k_t and working in $\mathbb{Z}$, we see that

$$1 \leqslant |j_2 k_{t_2} - j_1 k_{t_1}| \leqslant (s-1)\big(\lceil \sqrt{n} \rceil + \lceil n^{1/4} \rceil\big) < n,$$

and it follows that

$$j_1 k_{t_1} \neq j_2 k_{t_2} \pmod n. \tag{1}$$

Finally, suppose that some distinct β_1, β_2 satisfy $\{\beta_1, \beta_2\} \subset H_{t_1}(\alpha) \cap H_{t_2}(\alpha)$ where $k_{t_1} \neq k_{t_2}$. Then $\beta_2 - \beta_1 = j k_{t_1} \pmod n$ for some $j_1 \in \{1, \ldots, s-1\}$, and $\beta_2 - \beta_1 = j_2 k_{t_2} \pmod n$ for some $j_2 \in \{1, \ldots, s-1\}$, but this contradicts (1). Therefore, by definition of k_t, for every $\{\beta_1, \beta_2\} \subset \mathbb{Z}_n$, we have

$$\texttt{vis}(\beta_1, \beta_2) = |\{t \in \{1, 2, \ldots, n\} \mid \{\beta_1, \beta_2\} \subset H_t(\alpha) \text{ for some } \alpha \in \mathbb{Z}_n\}| \leqslant \mathcal{O}(n^{3/4}).$$

- Stationary Geometric Mobile Network. Consider an R-dimensional torus $\Gamma(n, R)$, which is a graph whose vertex set is the Cartesian product of $\mathbb{Z}_\ell^R = \mathbb{Z}_\ell \times \ldots \times \mathbb{Z}_\ell$, where $\ell = n^{1/R} \in \mathbb{Z}$, and two vertices $(x_1, \ldots, x_R)$ and $(y_1, \ldots, y_R)$ are connected if for some $j \in \{1, 2 \ldots, R\}$ $x_j = y_j \pm 1 \bmod n$ and for all $i \neq j$ we have $x_i = y_i$. Let π be the stationary distribution of the following random walk on $\Gamma(n, R)$: at each step, the walker stays at the current vertex with probability p, and otherwise chooses a neighbour randomly and moves to that neighbour. The transition probability from vertex u to a neighbouring vertex w is $(1-p)/(2R)$, where $2R$ is the degree of vertex u in $\Gamma(n, R)$. Now place n agents on vertices of $\Gamma(n, R)$ independently, each according to the distribution π. At each time step, each agent independently performs a step of the random walk described above (For random walks on a torus we refer the interested reader to [15]). For every pair of distinct agents a and b, let $d_t(a, b)$ denote the Manhattan distance (in Γ) of the locations of a and b at time t. For a given $r \geqslant 1$, we define the *communication graph process* $\{G_r^{(t)} \mid t = 0, 1, \ldots\}$ over the set of agents, say A, so that for every $t \geqslant 0$, agents a and b are connected if and only if $d_t(a, b) \leqslant r$. The model has been thoroughly studied when $R = 2$ in the context of information spreading [9]. We present the following result regarding the pair visibility of the communication graph process, proved in Appendix B.

▶ **Proposition 8.** *Fix $r = r(n) = n^{o(1)}$. Also let $\{G_r^{(t)} = (A, E_t) \mid 1 \leqslant t \leqslant n\}$ be the communication graph process defined on an R-dimensional torus $\Gamma(n, R)$. Then there exists constant $\varepsilon > 0$ such that for every pair of agents, say $\{a, b\} \subset A$,*

$$\texttt{vis}(a, b) = |\{t \in \{1, 2, \ldots, n\} \mid \{a, b\} \in E_t\}| = \mathcal{O}(n^{1-\varepsilon}).$$

1.2 Related Works

As we discussed, in the standard balls-into-bins, each ball picks a set of d choices from n bins, independently and uniformly at random. One of the first algorithms considering a different distribution over the bins is called *always-go-left* proposed by Vöcking [20]. In this algorithm, the bins are partitioned into d groups of size n/d and each ball picks one random bin from each group. The ball is then allocated to a least-loaded bin among the chosen bins, with ties broken in favor of the bin from the least-indexed group. The algorithm uses exponentially smaller number of choices and achieve a maximum load of $\frac{\log \log n}{d \phi_d} + \mathcal{O}(1)$,

where $1 \leqslant \phi_d \leqslant 2$ is an specified constant. Byers et al. [5] studied a model, where n bins are uniformly at random placed on a geometric space. Then each ball, in turn, picks d locations in the space. Corresponding to these d locations, the ball probes the load of d bins that have the minimum distance from the locations. The ball then allocates itself to one of the d bins with minimum load. In this scenario, the probability that a location close to a bin is chosen depends on the distribution of other bins in the space and hence there is not a uniform distribution over the potential choices. Here, the authors showed the maximum load is $\log_d \log n + \mathcal{O}(1)$. Later on, Kenthapadi and Panigrahy [13] proposed a graphical balanced allocation in which bins are interconnected as a s-regular graph and each ball picks a random edge of the graph. It is then placed in one of its endpoints with a smaller load. This allocation algorithm results in a maximum load of $\log \log n + \mathcal{O}\left(\frac{\log n}{\log(s/\log^4 n)}\right) + \mathcal{O}(1)$. Godfrey [12] studied balanced allocation on hypergraphs where each ball probes the bins contained in a random edge of size $\Omega(\log n)$. In [3, 12], the balanced allocation process on hypergraphs was studied where number of choices is $d = \Omega(\log n)$. The analysis involves the second moment method (Chernoff bounds), and lower bound on d is needed in order to achieve concentration. Hence it is unlikely that the techniques of [3,12] can be extended to the range $d = o(\log n)$. Peres et al. [17] also considered balanced allocation on graphs where the number of balls m can be much larger than n (i.e., $m \gg n$) and the graph is not necessarily regular and dense. Then, they established upper bound $\mathcal{O}(\log n/\sigma)$ for the gap between the maximum and the minimum loaded bin after allocating m balls, where σ is the edge expansion of the graph. Bogdan et al. [4] studied a model where each ball picks a random vertex and performs a local search from the vertex to find a vertex with local minimum load, where it is finally placed. They showed that when the graph is a constant degree expander, the local search guarantees a maximum load of $\Theta(\log \log n)$. Pourmiri [18] substitutes the local search by non-backtracking random walks of length $\ell = o(\log n)$ to sample the choices and then the ball is allocated to a least-loaded bin. Provided the underlying graph has sufficiently large girth and ℓ, he showed the maximum load is a constant. In the context of hashing (e.g., [1, 11]), authors apply the witness graph techniques to analyze the maximum load in the balls-into-bins process where the bins are picked based on tabulation.

2 Balanced Allocation on Dynamic Hypergraphs

In this section we establish an upper bound for the maximum load attained by the balanced allocation on hypergraphs (i.e., Theorem 2). In order to analyze the process let us first define a *conflict graph*. We write D_t for the set of d bins chosen by the t-th ball, and sometimes refer to D_t as the *d-choice* of the t-th ball. We will slightly abuse the notation and write $D_u \cap D_t$, $D_u \cup D_t$ to denote the set of common bins, and the union of bins, chosen by balls u and t, respectively.

▶ **Definition 9** (Conflict Graph). *For $m = 1, \ldots, n$, the conflict graph $\mathcal{C}_m$ is a simple graph with vertex set $\{D_1, D_2, \ldots, D_m\}$. Vertices D_u and D_t are connected by an edge in $\mathcal{C}_m$ if and only if $D_u \cap D_t \neq \emptyset$ (that is, the d-choices of the t-th ball and the u-th ball contain a common bin).*

We say a subgraph of $\mathcal{C}_m$ with vertex set $\{D_{t_1}, \ldots, D_{t_k}\}$ is *c-loaded* if every bin in $D_{t_1} \cup D_{t_2} \cup \cdots \cup D_{t_k}$ has at least c balls.

Our analysis will involve a useful combinatorial object, called an *ordered tree*. An ordered tree is a rooted tree, together with a specified ordering of the children of every vertex. Recall that $\frac{1}{k+1}\binom{2k}{k}$ is the k-th Catalan number, which counts numerous combinatorial objects,

including the number of ways to form k balanced parentheses. It is well known [19] that ordered trees with $k-1$ edges are counted by the $(k-1)$-th Catalan number, leading easily to the following proposition.

▶ **Proposition 10.** *The number of k-vertex ordered trees is $\frac{1}{k}\binom{2k-2}{k-1} \leqslant 4^{k-1}$.*

More information regarding the enumeration of trees can be found in [14].

The following blue-red coloring will be very helpful in our analysis.

▶ **Definition 11** (Blue-red coloring). *Given $m \in \{1, 2, \ldots, n\}$, suppose that $T \subset \mathcal{C}_m$ is a rooted and ordered k-vertex tree contained in $\mathcal{C}_m$. Let the vertex set of T be $\{D_{t_1} \ldots, D_{t_k}\}$, where D_{t_1} is the root. Perform depth-first search starting from the root, respecting the specified order of each vertex. For $i = 1, \ldots, k$, let $D(i) \in \{D_{t_1} \ldots, D_{t_k}\}$ be the vertex which is the i-th visited vertex in the depth-first search. Then $D(1) = D_{t_1}$ is the root. for $j = 1, \ldots, k$. We now define a blue-red coloring $col : \{D(2), \ldots, D(k)\} \to \{blue, red\}$ as follows. For $i = 2, \ldots, k$,*

$$col(D(i)) = \begin{cases} blue & \text{if } |(\cup_{j=1}^{i-1} D(j)) \cap D(i)| = 1, \\ red & \text{if } |(\cup_{j=1}^{i-1} D(j)) \cap D(i)| \geqslant 2. \end{cases}$$

The following key lemma presents a upper bound for the probability that a certain tree can be found as a subgraph of $\mathcal{C}_m$.

▶ **Lemma 12** (Key Lemma). *Let $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ be a dynamic s-uniform hypergraph which satisfies the β-balanced, ε-visibility and c_0-size properties. Suppose that $c > 0$ is an arbitrary constant and $k = C \log n$ for some constant $C \geqslant 1$. There exists $\Theta(n) \leqslant m \leqslant n$ such that the probability that $\mathcal{C}_m$ contains a c-loaded k-vertex tree with r red vertices in its blue-red colouring is at most*

$$n^{c_0+3} \exp\{4k \log(2\beta d) - r\varepsilon \log(n)/2 - c(d-1)(k-r-1)\}.$$

Moreover, with high probability, $r = \mathcal{O}(1/\varepsilon)$.

The proof, presented in Section 3, involves an extension of the witness tree technique. This method might be of independent interest in the study of random hypergraphs.

We now explain how to recursively build a witness graph if there exists a bin whose load is higher than a certain threshold. The *minimum load* of D_t is the number of balls in the least-loaded bin in D_t (the set of d choices of D_t). Clearly, if ball t is placed at height h then D_t has minimum load at least h.

Construction of the Witness Graph

Suppose that there exists a bin with load $\ell + c + 1$. Let R be the d-choice corresponding to the ball at height $\ell + c$ in this bin. Then the minimum load of R is $\ell + c$. We start building the witness tree in $\mathcal{C}_m$ whose root is R. For every bin $i \in R$, consider the ℓ balls in bin i at height $\ell + c - j$, for $j = 1, \ldots, \ell$, and let $D^i_{t_j}$ be the d-choice corresponding to the ball in bin i with height $\ell + c - j$. These ℓ balls exist as the minimum load of R is $\ell + c$. We refer to set $\{D^i_{t_j} \mid i \in R,\ 1 \leqslant j \leqslant \ell\}$ as the set of *children* of R, where the minimum load of $D^i_{t_j}$ is $\ell + c - j - 1$. All children of R are connected to R in $\mathcal{C}_m$. Order the children of R arbitrarily, then blue-red colour the first level of the tree (the children of R). Recall that a vertex is colored by blue if it only shares one bin with its predecessors in the ordering. So a blue d-choice contains $d-1$ bins that have not appeared in previous d-choices (with respect to depth-first search, respecting the fixed ordering). We call these $d-1$ bins *fresh*.

Next, consider each blue vertex of the tree (if any), and recover the d-choices corresponding to balls that are placed in fresh bins with height at least c. Then, blue-red color the children of those d-choices, with respect to an arbitrary ordering. This recursion will continue until either there are no balls remaining with height at least c, or there are no blue vertices. For $j = 1, \dots, \ell$, let $f(\ell - j)$ denote the number of d-choices that the recursive construction gives, when the d-choice for the root has minimum load $\ell + c - j - 1$. Provided all vertices are colored blue, the recursive construction continues until no ball remains with height at least c. Therefore, a simple calculation shows that

$$f(\ell) \geqslant (d-1)(f(\ell-1) + f(\ell-2) + \cdots + f(0) + 1),$$

where $f(0) = 1$. Solving the above recursive formula shows that $f(\ell) \geqslant 2(d-1)d^{\ell-1} \geqslant d^{\ell}$.

Proof of Theorem 2. Let $(\mathcal{H}^{(1)}, \dots, \mathcal{H}^{(n)})$ be a dynamic hypergraph which satisfies the β-balanced, ε-visibility and c_0-size properties. By Lemma 12, there exists $\Theta(n) = m \leqslant n$ such that the following holds with high probability: after m balls have been allocated by the balanced allocation process, if $T \subseteq \mathcal{C}_m$ is a c-loaded tree with k vertices and T is blue-red coloured according to some arbitrary ordering of the children of each vertex, then the number r of red vertices satisfies $r = \mathcal{O}(1/\varepsilon)$. So we are able to find a constant $c_2 \geqslant 0$ such that, with high probability, $r < c_2 \cdot d$.

Now suppose that after allocating m balls, there is a ball at height $\ell + c_1 + c_2 + 1$. This implies that there is a d-choice, denoted by R, whose minimum load is at least $\ell + c_1 + c_2 + 1$. Let us consider all balls placed in the bins contained in R with height at least $\ell + c_1 + 1$. Recover the corresponding d-choices for these balls, say $D_1, D_2, \dots, D_w$, then colour them blue-red with respect to the root R and an arbitrary ordering of the children of each vertex. Since $w \geqslant c_2 \cdot d$, w.h.p., there are $b \geqslant 1$ blue vertices and $w - b$ red vertices. We now consider every blue vertex $D_t \in \{D_1, D_2, \dots, D_w\}$ as a root and start the recursive construction of the witness graph. Assuming that the number of red vertices is strictly less than $c_2 \cdot d < w$, it follows that at least one recursive construction (with root D_i) does not produce any red vertex. Moreover, the recursion from D_i gives a c_1-loaded tree with at least $k = d^{\ell}$ vertices. We take $\ell = \log_d \log n$, so that $k = \log n$. Another application of Lemma 12 implies that a c_1-loaded k-vertex tree with no red vertices exists with probability at most

$$\begin{aligned} n^{c_0+3} \exp\{4k \log(2\beta d) - c_1(d-1)(k-1)\} &\leqslant \exp\left\{\left(c_0 + 4 + 4\log(2\beta d) - c_1(d-1)\right)\log n\right\} \\ &\leqslant \exp\left\{\left(c_0 + 4 + 4\log(4\beta) - c_1\right)\log n\right\}, \end{aligned}$$

using the fact that $2 \leqslant d = o(\log n)$ and $k = \log n$. Setting c_1 to be a large enough positive constant, we conclude that with high probability the maximum load is at most

$$\log_d \log n + \mathcal{O}(1) + c_2 = \log_d \log n + \mathcal{O}(1/\varepsilon),$$

where $c_2 = \mathcal{O}(1/\varepsilon)$. This proves the first statement of Theorem 2.

In order to prove the second statement of Theorem 2 we show the sub-additivity of the balanced allocation algorithm. We want to prove that for every constant integer $\gamma \geqslant 1$ with $\gamma m \leqslant n$, after allocating γm balls, the maximum load is at most $\gamma(\log_d \log n + \mathcal{O}(1/\varepsilon))$, with high probability. First assume that $2m \leqslant n$ and suppose that the algorithm has allocated m balls to $\mathcal{H}^{(t)}, t = 1, \dots, m$ and let $\ell^* \leqslant \log_d \log n + \mathcal{O}(1)$ denote its maximum load. We now consider two independent balanced allocation algorithms, say $\mathcal{A}$ and $\mathcal{A}_0$, on two dynamic hypergraphs starting from step m. These dynamic hypergraphs are $(\mathcal{H}^{(m)}, \dots, \mathcal{H}^{(n)})$ and $(\mathcal{H}_0^{(m)}, \dots, \mathcal{H}_0^{(n)})$, where $\mathcal{H}_0^{(t)}$ is an identical copy of $\mathcal{H}^{(t)}$ for $t = m, \dots, n$. Moreover, we assume that in round m, all bins contained in $\mathcal{H}_0^{(m)}$ have exactly ℓ^* balls. Let us couple

algorithm $\mathcal{A}$ on $\mathcal{H}^{(t)}$ and algorithm $\mathcal{A}_0$ on $\mathcal{H}_0^{(t)}$. Write $V = [n]$ for the set of n bins. To do so, the coupled process allocates a pair of balls to bins as follows: for $t = m+1, \ldots, 2m$, the coupling chooses a one-to-one labeling function $\sigma_t : V \to \{1, 2, \ldots, n\}$ uniformly at random, where V is the ground set of both hypergraphs (i.e, set of n bins) and $\{1, 2, \ldots, n\}$ is a set of labels. Next, the coupling chooses D_t randomly from $\mathcal{H}^{(t)}$. Let D'_t denote the same set of d bins as D_t in $\mathcal{H}_0^{(t)}$. Algorithm $\mathcal{A}$ allocates ball $t+1$ to a least-loaded vertex of D_t, and algorithm $\mathcal{A}_0$ allocates ball $t+1$ to a least-loaded vertex of D'_t, with both algorithms breaking ties in favour of the vertex v with the smallest load and minimum label $\sigma_t(v)$. Note that algorithm $\mathcal{A}$ is a faithful copy of the balanced allocation process on $(\mathcal{H}^{(m)}, \ldots, \mathcal{H}^{(n)})$, and algorithm $\mathcal{A}_0$ is a faithful copy of the balanced allocation process on $(\mathcal{H}_0^{(m)}, \ldots, \mathcal{H}_0^{(n)})$, respectively. (This follows as σ_t is chosen uniformly at random.) Let X_i^t and Y_i^t, $m+1 \leqslant t \leqslant 2m$, denote the load of bin i in $\mathcal{H}^{(t)}$ and $\mathcal{H}_0^{(t)}$, respectively. We prove by induction that for every integer $m \leqslant t \leqslant 2m$ and $i \in V$ we have

$$X_i^t \leqslant Y_i^t. \tag{2}$$

The inequality holds when by the assumption that $Y_i^m = \ell^*$ for every $i \in V$. Let us assume that for every t', $t' \leqslant t \leqslant 2m$, Inequality (2) holds, then we will show it for $t+1$. Let $i \in D_{t+1}$ and $j \in D'_{t+1}$ denote the vertices (bins) that receive a ball in step $t+1$. We now consider two cases:

- **Case 1:** $X_i^t < Y_i^t$. Since algorithm $\mathcal{A}$ allocated ball $t+1$ to bin i, it follows that

  $$X_i^t + 1 = X_i^{t+1} \leqslant Y_i^t \leqslant Y_i^{t+1}.$$

  So, Inequality (2) holds for $t+1$ and every bin $i \in V$.
- **Case 2:** $X_i^t = Y_i^t$. Since D'_{t+1} is a copy of D_{t+1}, we have $j \in D_{t+1}$ and $i \in D'_{t+1}$. We know that no vertex (bin) in D_{t+1} has smaller load than i, and no vertex (bin) in D'_{t+1} has smaller load than j. Hence

  $$X_i^t \leqslant X_j^t \leqslant Y_j^t \leqslant Y_i^t,$$

  where the middle inequality follows from the inductive hypothesis (2) for bin j. So by assumption of this case we obtain $X_i^t = X_j^t = Y_j^t = Y_i^t$. If $i \neq j$ and $\sigma_{t+1}(j) < \sigma_{t+1}(i)$, then it contradicts the fact that ball $t+1$ is allocated to bin i by algorithm $\mathcal{A}$. Similarly, if $\sigma_{t+1}(j) > \sigma_{t+1}(i)$, then it contradicts the fact that algorithm $\mathcal{A}_0$ allocated ball t to bin j. Therefore $i = j$ and hence

  $$X_i^{t+1} = X_i^t + 1 = Y_i^t + 1 = Y_i^{t+1}.$$

Thus, in both cases, Inequality (2) holds for every $t \geqslant 0$. By applying the first part of the theorem, with high probability, using algorithm $\mathcal{A}_0$ to allocate m balls to the dynamic hypergraph $(\mathcal{H}_0^{(m)}, \ldots, \mathcal{H}_0^{(2m)})$ results in maximum load

$$\ell^* + \log_d \log n + \mathcal{O}(1/\varepsilon) \leqslant 2(\log_d \log n + \mathcal{O}(1/\varepsilon))$$

in $\mathcal{H}_0^{(2m)}$. Therefore, by Inequality (2), after using algorithm $\mathcal{A}$ to allocate m balls to the dynamic hypergraph $(\mathcal{H}^{(m)}, \ldots, H^{(n)})$, with high probability the maximum load in $\mathcal{H}^{(2m)}$ is at most $2(\log_d \log n + \mathcal{O}(1/\varepsilon))$. Applying the union bound, we conclude that after allocating γm balls, where $\gamma m \leqslant n$, the maximum load is at most $\gamma(\log_d \log n + \mathcal{O}(1/\varepsilon))$, with high probability. This completes the proof of Theorem 2. ◀

3 Appearance Probability of a Certain Structure

In this section we work towards a proof of Lemma 12. First we will give some useful definition and prove some helpful results. The definition was introduced in [18].

▶ **Definition 13.** *Suppose that $\mathcal{A}$ is an allocation algorithm that sequentially allocates n balls into n bins according to some mechanism. For a given constant $\alpha > 0$, and for $\Theta(n) = m \leqslant n$, we say that $\mathcal{A}$ is (α, m)-uniform if for every ball $1 \leqslant t \leqslant m = \Theta(n)$ and every bin $i \in [n]$,*

$$\mathbf{Pr}\left[\text{ ball } t \text{ is allocated to bin } i \text{ by } \mathcal{A} \mid \text{balls } 1, 2, \ldots, t-1 \text{ have been allocated by } \mathcal{A}\right] \leqslant \frac{\alpha}{n}.$$

In the above definition, we condition on the allocations of balls $1, \ldots, t-1$ into bins made by $\mathcal{A}$.

The following result will be proved in Appendix C.

▶ **Lemma 14.** *Fix $d = d(n)$ with $2 \leqslant d = o(\log n)$. Suppose that the dynamic s-uniform hypergraph $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ satisfies the balancedness and size properties. There exists $m = \Theta(n)$ with $m < n$ such that with probability at least $1 - n^{-2}$, the edge H_t chosen by the t-th ball contains at least $s/2$ empty vertices for $t = 1, \ldots, m$.*

Using this result we can prove that the balanced allocation process is uniform on dynamic hypergraphs.

▶ **Lemma 15** (Uniformity Lemma)**.** *Fix $d = d(n)$ with $2 \leqslant d = o(\log n)$ and suppose that for some constant $\beta \geqslant 1$, the s-uniform dynamic hypergraph $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ satisfies the β-balanced and size properties, with $d \leqslant s$. Then there exists a constant $\alpha = \alpha(\beta)$, which depends only on β, and there exists $m = \Theta(n)$ with $m < n$, such that the balanced allocation process on $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ is (α, m)-uniform. Specifically, we may take $\alpha = 44\beta$.*

Proof. Fix $m = m(n)$ to equal the m provided by Lemma 14. For $t = 1, \ldots, m$, let D_t be the d-element subset of H_t that is chosen by the t-th ball. Define the indicator random variable $\mathbb{I}_t$ as follows:

$$\mathbb{I}_t := \begin{cases} 1 & \text{if } D_t \text{ contains at least } d/6 \text{ empty vertices,} \\ 0 & \text{otherwise.} \end{cases}$$

Let us fix an arbitrary bin i and then define $A(t, i)$ to be the event that the t-th ball is allocated to vertex i. (The first $t-1$ balls have already been allocated, as the balanced allocation process never fails.) Observe that if $i \notin D_t$ then $\mathbf{Pr}\left[A(t,i)\right] = 0$. It follows that

$$\begin{aligned} \mathbf{Pr}\left[A(t,i)\right] &= \mathbf{Pr}\left[A(t,i) \mid i \in D_t \text{ and } \mathbb{I}_t = 1\right] \cdot \mathbf{Pr}\left[i \in D_t \text{ and } \mathbb{I}_t = 1\right] \\ &\quad + \mathbf{Pr}\left[A(t,i) \mid i \in D_t \text{ and } \mathbb{I}_t = 0\right] \cdot \mathbf{Pr}\left[i \in D_t \text{ and } \mathbb{I}_t = 0\right]. \end{aligned}$$

Now there are at least $d/6$ empty vertices on D_t when $\mathbb{I}_t = 1$, so

$$\mathbf{Pr}\left[A(t,i) \mid i \in D_t \text{ and } \mathbb{I}_t = 1\right] \leqslant 6/d.$$

It follows that

$$\begin{aligned} \mathbf{Pr}\left[A(t,i)\right] &\leqslant (6/d)\ \mathbf{Pr}\left[i \in D_t \text{ and } \mathbb{I}_t = 1\right] + \mathbf{Pr}\left[i \in D_t \text{ and } I_t = 0\right] \\ &\leqslant (6/d)\ \mathbf{Pr}\left[i \in D_t\right] + \mathbf{Pr}\left[\mathbb{I}_t = 0 \mid i \in D_t\right] \cdot \mathbf{Pr}\left[i \in D_t\right]. \end{aligned} \tag{3}$$

In order to have $i \in D_t$, first an edge containing i must be selected, and then the chosen d-element subset of that edge must contain i. By the β-balancedness property,

$$\mathbf{Pr}\left[i \in D_t\right] \leqslant \frac{\beta s}{n} \cdot \frac{\binom{s-1}{d-1}}{\binom{s}{d}} \leqslant \frac{\beta}{n}.$$

Using the above inequality, we simplify Inequality (3) as follows:

$$\mathbf{Pr}\left[A(t,i)\right] \leqslant \frac{6\beta}{n} + \frac{\beta d}{n}\,\mathbf{Pr}\left[\mathbb{I}_t = 0 \mid i \in D_t\right].$$

If $d \leqslant 6$ then the above inequality immediately implies that $\mathbf{Pr}\left[A(t,i)\right] \leqslant 12\beta/n$. This completes the proof when $d \leqslant 6$. For the remainder of the proof we assume that $d \geqslant 7$, and prove that

$$\mathbf{Pr}\left[\mathbb{I}_t = 0 \mid i \in D_t\right] \leqslant \hat{c}/d \tag{4}$$

for some absolute constant $\hat{c} > 0$. From this, we see that $\mathbf{Pr}\left[A(t,i)\right] \leqslant \alpha/n$ where $\alpha = \beta(6+\hat{c})$. As i was an arbitrary bin, this proves that the process is (α, m)-uniform.

Let $\mathcal{F}$ be the event that H_t contains at least $s/2$ empty vertices for all $t = 1, \ldots, m$. By Lemma 14, we have $\mathbf{Pr}\left[\mathcal{F}\right] \geqslant 1 - n^{-2}$. Then

$$\begin{aligned}
&\mathbf{Pr}\left[\mathbb{I}_t = 0 \mid i \in D_t\right] \\
&= \mathbf{Pr}\left[\mathbb{I}_t = 0 \mid (i \in D_t) \text{ and } \mathcal{F}\right] \cdot \mathbf{Pr}\left[\mathcal{F}\right] + \mathbf{Pr}\left[\mathbb{I}_t = 0 \mid (i \in D_t) \text{ and } \neg\mathcal{F}\right] \cdot \mathbf{Pr}\left[\neg\mathcal{F}\right] \\
&\leqslant \mathbf{Pr}\left[\mathbb{I}_t = 0 \mid (i \in D_t) \text{ and } \mathcal{F}\right] + n^{-2} \leqslant \mathbf{Pr}\left[\mathbb{I}_t = 0 \mid (i \in D_t) \text{ and } \mathcal{F}\right] + 1/d.
\end{aligned} \tag{5}$$

Let X be the random variable that counts the number of empty bins of a random $(d-1)$-element subset of $H_t \setminus \{i\}$, conditioned on the event that "$(i \in D_t)$ and $\mathcal{F}$" holds. Then X is a hypergeometric random variable with parameters $(s-1, K, d-1)$, where K is the number of empty bins contained in $H_t \setminus \{i\}$. Thus

$$\mathbf{E}\left[X\right] = \frac{(d-1)K}{s-1} \quad \text{and} \quad \mathbf{Var}\left[X\right] \leqslant \frac{(d-1)K}{s-1} \leqslant d.$$

Then $\mathbf{E}\left[X\right] > d/3$, since $K \geqslant s/2 - 1$ when $i \in D_t$ and $\mathcal{F}$ holds (and using the size property $s = \Omega(\log n)$) and the fact that $d \geqslant 7$). Therefore

$$\begin{aligned}
&\mathbf{Pr}\left[\mathbb{I}_t = 0 \mid (i \in D_t) \text{ and } \mathcal{F}\right] \\
&\leqslant \mathbf{Pr}\left[X < d/6\right] \leqslant \mathbf{Pr}\left[\left|X - \mathbf{E}\left[X\right]\right| \leqslant \mathbf{E}\left[X\right]/2\right] < \frac{4\,\mathbf{Var}\left[X\right]}{\mathbf{E}\left[X\right]^2} \leqslant \frac{36d}{d^2} = \frac{36}{d},
\end{aligned}$$

using Chebychev's inequality. Substituting the above upper bound in Inequality (5) establishes (4) with $\hat{c} = 38$, which completes the proof. ◀

We are ready to prove Lemma 12.

▶ **Lemma 16** (Restatement of Lemma 12). *Fix $d = d(n)$ with $2 \leqslant d = o(\log n)$. Let $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ be a dynamic hypergraph which satisfies the β-balanced, ε-visibility and c_0-size properties. Suppose that $c \geqslant 44\beta\mathrm{e}^2$ is a sufficiently large constant, and let $k = C\log n$ for some constant $C \geqslant 1$. There exists $\Theta(n) \leqslant m \leqslant n$ such that the probability that $\mathcal{C}_m$ contains a c-loaded k-vertex tree is at most*

$$\exp\left\{4k\log(2\beta d) - c(d-1)(k-r-1) + \left(c_0 + 3 - r\varepsilon/2\right)\log(n)\right\}$$

where r is the number of red vertices in the blue-red coloring of the tree. Moreover, with high probability, if $\mathcal{C}_m$ contains any such tree then $r = \mathcal{O}(1/\varepsilon)$.

Proof. Fix $m = m(n)$ to equal the m provided by Lemma 15. There are at most 4^k ordered trees with k vertices. (Proposition 10). Fix such a tree, say T, and label the vertices $\{1, 2, \ldots, k\}$ such that vertex i is the i-th new vertex visited when performing depth-first search in T starting from the root, and respecting the given ordering. In particular, the root of T is vertex 1. Next, we will assign a d-choice to the root vertex of T, as a first step in describing trees which may be present in the witness graph $\mathcal{C}_m$. Let x count the number of possible d-choices that can be assigned to the root of T. Then

$$x \leqslant \binom{s}{d} \cdot \left| \bigcup_{t=1}^{m} \mathcal{E}_t \right| \cdot m \leqslant \binom{s}{d} \cdot n^{c_0+2},$$

where the last inequality follows from the size property and the inequality $m \leqslant n$. Therefore, there are x possibilities for the root and hence there are at most $4^k \cdot \binom{s}{d} \cdot n^{c_0+2}$ ordered trees with the specified root. Fix an arbitrary d-choice D_t as the root for T.

Next we fix an arbitrary function $\mathrm{col} : \{2, \ldots, k\} \to \{\text{blue}, \text{red}\}$, that gives a blue-red coloring of $2, \ldots, k$. In what follows we establish an upper bound for the probability that $\mathcal{C}_m$ contains the blue-red colored tree $T \subset \mathcal{C}_m$, (according to Definition 11). Let $q_1(t)$ be the probability that the t-th ball chooses the root of T (that is, that the d-choice made by the t-th ball corresponds to the root of T). Then

$$\sum_{t=1}^{m} q_1(t) \leqslant \sum_{t=1}^{m} \frac{1}{\binom{s}{d}} \leqslant \frac{n}{\binom{s}{d}}, \tag{6}$$

because H contains $\binom{s}{d}$ distinct d-element sets for for every $H \in \mathcal{E}_t$. For every $t = 2, \ldots, k$, define $q_i(t, \mathrm{col}(i))$ to be the probability that the t-th ball chooses the i-th vertex of the tree (i.e., i) with $\mathrm{col}(i)$. If $\mathrm{col}(i)$ is red then D_t must share at least two bins with $\cup_{j=1}^{i-1} D_{t_j}$, while if $\mathrm{col}(i)$ is blue then D_t only shares one bin with its parent. For every $i = 2, \ldots, k$, let us derive an upper bound on $q_i(t, \text{blue})$. Here, the i-th vertex share one bin with its parent in T, say D_{t_j}. Now D_{t_j} has d bins and by the balancedness property we get

$$\mathbf{Pr}\left[D_{t_j} \cap H_t \neq \emptyset\right] \leqslant \sum_{i \in D_{t_j}} \mathbf{Pr}\left[i \in H_t\right] \leqslant \frac{\beta d s}{n},$$

where H_t is the edge chosen by ball t from $\mathcal{H}^{(t)}$, uniformly at random. Suppose that for some $a \geqslant 1$ we have $|D_{t_j} \cap H_t| = a \leqslant d$. Then the total number of d-element subsets of H_t which share only one bin with D_{t_j} is $a\binom{s-a}{d-1} \leqslant d\binom{s-1}{d-1}$. Thus, we get

$$\sum_{t=1}^{m} q_i(t, \text{blue}) \leqslant \sum_{t=1}^{m} \frac{\beta d s}{n} \cdot d \frac{\binom{s-1}{d-1}}{\binom{s}{d}} = \sum_{t=1}^{m} \frac{\beta d^3}{n} \leqslant \beta d^3, \tag{7}$$

because $m \leqslant n$.

Next, for every $i = 2, \ldots, k$, and every $t = 2, \ldots, m$, we need an upper bound on $q_i(t, \text{red})$. If the i-th vertex of the tree is the set D_t and is coloured red, then D_t is a d-element set of bins which shares at least two bins with $\cup_{j=1}^{i-1} D_{t_j}$. One of these bins belongs to the (known) parent, and the other belongs to $D_{t_1} \ldots, D_{t_{i-1}}$. So if U is the number of choices for this pair of bins, then

$$U \leqslant d \cdot (i-1)d \leqslant kd^2. \tag{8}$$

Let $\{p_1, p_2 \ldots, p_U\}$ be the set of such pairs of bins. For $J = 1, \ldots, U$, write $A(p_J, t)$ for the event that the pair p_J is contained in a randomly chosen edge of $\mathcal{E}_t$. Observe that if $p_J \subset D_t$ then $A(p_J, t)$ holds. Then, by the balancedness property we have

$$
\begin{aligned}
&\mathbf{Pr}\left[p_J \subset D_t\right] \\
&= \mathbf{Pr}\left[p_J \subset D_t \mid A(p_J,t)\right] \cdot \mathbf{Pr}\left[A(p_J,t)\right] \\
&\leqslant \mathbf{Pr}\left[p_J \subset H_t\right] \cdot \frac{\binom{s-2}{d-2}}{\binom{s}{d}} \cdot \mathbf{Pr}\left[A(p_J,t)\right] \\
&\leqslant \mathbf{Pr}\left[p_J \cap H_t \neq \emptyset\right] \cdot \frac{\binom{s-2}{d-2}}{\binom{s}{d}} \cdot \mathbf{Pr}\left[A(p_J,t)\right] \\
&\leqslant \frac{2\beta s}{n} \cdot \frac{\binom{s-2}{d-2}}{\binom{s}{d}} \cdot \mathbf{Pr}\left[A(p_J,t)\right] = \frac{2\beta d(d-1)}{(s-1)n}\ \mathbf{Pr}\left[A(p_J,t)\right],
\end{aligned}
$$

as $\binom{s-2}{d-2}$ is the number of d-element subsets of H_t which contain the pair p_J. Then

$$
q_i(t,\mathrm{red}) \leqslant \sum_{J=1}^{U} \frac{2\beta d(d-1)}{(s-1)n}\ \mathbf{Pr}\left[A(p_J,t)\right].
$$

Note that by (8) we have $U \leqslant kd^2$ and hence,

$$
\sum_{t=1}^{m} q_i(t,\mathrm{red}) \leqslant \sum_{J=1}^{U}\sum_{t=1}^{n} \frac{2\beta d(d-1)}{(s-1)n}\ \mathbf{Pr}\left[A(p_J,t)\right] \leqslant \sum_{J=1}^{U} \frac{2\beta d(d-1)}{(s-1)n}\ \mathtt{vis}(p_J) \leqslant \frac{2\beta k d^4}{n^{\varepsilon}}. \tag{9}
$$

The final inequality follows from the visibility property, using the fact that $d < s$.

Write $\mathrm{col}^{-1}(\mathrm{blue})$ for the set of blue vertices in T, and similarly for $\mathrm{col}^{-1}(\mathrm{red})$. Then

$$
|\mathrm{col}^{-1}(\mathrm{red})| + |\mathrm{col}^{-1}(\mathrm{blue})| = k-1.
$$

Suppose that $(t_1,\ldots,t_k)$ is the sequence of balls that are going to select vertices $1,2,\ldots,k$ of T. By applying (6), (7) and (9), we find that the probability that the edges of the colored tree T appears in $\mathcal{C}_m$ at times $(t_1,\ldots,t_k)$, and the corresponding sets $D_{t_1},\ldots,D_{t_k}$ consistent with chosen blue-red coloring scheme, is at most

$$
\begin{aligned}
&\sum_{(t_1,\ldots,t_k)} \left\{ q_1(t_1) \prod_{i=2}^{k} q_i(t_i,\mathrm{col}(i)) \right\} \leqslant \left(\sum_{t=1}^{m} q_1(t) \right) \prod_{2=1}^{k} \left(\sum_{t=1}^{m} q_i(t,\mathrm{col}(i)) \right) \\
&\leqslant \frac{n}{\binom{s}{d}} \left(\prod_{i\in\mathrm{col}^{-1}(\mathrm{blue})} \sum_{t=1}^{m} q_i(t,\mathrm{blue}) \right) \left(\prod_{i\in\mathrm{col}^{-1}(\mathrm{red})} \sum_{t=1}^{m} q_i(t,\mathrm{red}) \right) \\
&\leqslant \frac{n}{\binom{s}{d}} \left(\beta d^3\right)^{|\mathrm{col}^{-1}(\mathrm{blue})|} \left(\frac{2\beta k d^4}{n^{\varepsilon}} \right)^{|\mathrm{col}^{-1}(\mathrm{red})|} \leqslant \frac{n\beta^k d^{4k}}{\binom{s}{d}} \left(\frac{2k}{n^{\varepsilon}} \right)^{|\mathrm{col}^{-1}(\mathrm{red})|}.
\end{aligned} \tag{10}
$$

There are at most 2^{k-1} coloring functions and $4^k \mathrm{poly}(n)\binom{s}{d}$ rooted and ordered trees. So by the upper bound (10), together with the union bound over all colored ordered trees, we obtain

$$
\begin{aligned}
&\mathbf{Pr}\left[\ \mathcal{C}_m \text{ contains a valid blue-red colored } k\text{-vertex tree with } r \text{ red vertices }\right] \\
&\leqslant 4^k 2^{k-1} \cdot n^{c_0+2} \binom{s}{d} \cdot \frac{n\,\beta^k d^{4k}}{\binom{s}{d}} \left(\frac{2k}{n^{\varepsilon}} \right)^{r} \\
&\leqslant n^{c_0+3} \cdot (2\beta d)^{4k} \cdot n^{-r\varepsilon/2} \leqslant \exp\left(4k\log(2\beta d) + (c_0+3-r\varepsilon/2)\log n\right),
\end{aligned} \tag{11}
$$

using $k = \mathcal{O}(\log n)$ for the penultimate inequality.

Let $b = k - r - 1$ be the number of blue vertices and let $D_{s_1}, \dots, D_{s_b}$ be the sorted list of blue vertices such that $s_1 < s_2 < \cdots < s_b$. Then, by the definition of blue-red coloring, for every $j = 1, \dots, b$ we have $|(\cup_{g=1}^{j-1} D_{s_g}) \cap D_{s_j}| \leqslant 1$. This implies that

$$y = |\cup_{j=1}^{k} D_{t_j}| \geqslant |\cup_{j=1}^{b} D_{s_j}| \geqslant (d-1)b = (d-1)(k-1-r),$$

since $\{s_1, \dots, s_b\} \subseteq \{t_1, \dots, t_k\}$. Applying Lemma 15 implies that the balanced allocation is (α, m)-uniform, where $\alpha = 44\beta$, say. Hence for any $c \geqslant 44\beta \mathrm{e}^2$, the probability that each bin in $\cup_{j=1}^{k} D_{t_j}$ is allocated at least c balls (that is, the tree T is c-loaded) is at most

$$\binom{m}{cy} \left(\frac{\alpha y}{n}\right)^{cy} \leqslant \left(\frac{\mathrm{e}m}{cy}\right)^{cy} \left(\frac{\alpha y}{n}\right)^{cy} \leqslant \left(\frac{\mathrm{e}\alpha}{c}\right)^{cy} \leqslant \mathrm{e}^{-c(d-1)(k-r-1)},$$

where the last inequality follows from $m \leqslant n$ and the fact that $c > \alpha \mathrm{e}^2$. Since balls are independent from each other, we can multiply the above inequality by (11) to show that the probability that $\mathcal{C}_m$ contains a c-loaded k-vertex tree with r red vertices is at most

$$\exp\Big\{4k\log(2\beta d) - c(d-1)(k-r-1) + \big(c_0 + 3 - r\varepsilon/2\big)\log n\Big\}, \tag{12}$$

proving the first statement of the lemma. Finally, suppose that $r\varepsilon \to \infty$ as $n \to \infty$. Then the upper bound in (12) can be written as

$$\begin{aligned}&\exp\Big\{\big(4\log(2\beta d) - c(d-1)\big)k + \mathcal{O}(\log n) + o(r \cdot \log n) - (r\varepsilon/2)\log n\Big\}\\ &\leqslant \exp\Big\{\mathcal{O}(\log n) + o(r \cdot \log n) - (r\varepsilon/2)\log n\Big\}.\end{aligned}$$

Since $r\varepsilon \to \infty$, this term dominates and the probability that $\mathcal{C}_m$ contains a blue-red coloured tree with r red vertices tends to zero. Therefore, if such a tree is present in $\mathcal{C}_m$ then $r = \mathcal{O}(1/\varepsilon)$ with high probability. This completes the proof. ◀

4 Balanced Allocation on Dynamic Graphs

In this section we show an upper bound for maximum load attained by the balanced allocation on regular dynamic graphs (i.e., Theorem 6). Suppose that the balanced allocation process has allocated n balls to the dynamic regular graph $(G^{(1)}, \dots, G^{(n)})$. Define the *conflict graph* $\mathcal{C}_n$ formed by the edges selected by the n balls. The vertex set of $\mathcal{C}_n$ is the set $[n]$ of bins, and the loads of these bins are updated during the process.

Given a tree T which is a subgraph of $\mathcal{C}_n$, and vertices u, v of the tree, if $\{u, v\}$ is an edge of $\mathcal{C}_n$ then we say it is a *cycle-producing edge* with respect to the tree T. The name arises as adding this edge to the tree would produce a cycle, which may be a 2-cycle if the edge $\{u, v\}$ is already present in T. For a positive integer $c > 0$, a subgraph of $\mathcal{C}_n$ is called *c-loaded* if each vertex (bin) contained in the subgraph has load at least c. The following proposition presents some properties of connected components of $\mathcal{C}_n$.

▶ **Proposition 17.** *Let $(G^{(1)}, \dots, G^{(n)})$ be a regular dynamic graph on vertex set $[n]$ which is ε-visible. Let that $\mathcal{C}_n$ be the conflict graph obtained after allocating n balls using the balanced allocation process. Then for every given constant $c > 0$, with probability at least $1 - n^{-c}$, every $12(c+1)$-loaded connected component of $\mathcal{C}_n$ contains strictly fewer than $\log n$ vertices. Moreover, the number of cycle-producing edges in the component is at most $2(c+1)/\varepsilon$.*

We will prove the proposition in Appendix D. We now explain how to recursively build a witness graph, provided there exists a bin whose load is higher than a certain threshold.

Construction of the Witness Graph

Let us start with a bin, say r, with $\ell + c$ balls. Clearly, if a ball is in bin r at height h then the other bin it chose, as part of the balanced allocation procedure, had load at least h. Starting from bin (vertex) r, let us recover all ℓ edges corresponding to the balls that were placed in r with height at least c. Thus, the alternative bin choices have loads at least $\ell + c - 1, \ldots, c$, respectively. These ℓ bins are all neighbours of r in $\mathcal{C}_n$, and we refer to them as the children of r. Next, we recover the edges corresponding to balls placed in the children of r at height at least c. Recursively, we continue until there is no ball remaining at height c or more. For every $i = 1, \ldots, \ell$, let $f(\ell - i)$ denote the number of vertices generated by the recursive construction, starting with a bin which contains $\ell - i + c$ balls. Assume for the moment that, for each vertex with load at least c, the recursive procedure always gives produces distinct children. Then

$$f(\ell) \geqslant f(\ell - 1) + f(\ell - 2) + \ldots + f(0) + 1,$$

where $f(0) = 1$. A simple calculation shows that $f(\ell) \geqslant 2^\ell$. Thus, the recursive procedure gives a c-loaded tree with at least 2^ℓ vertices, under the assumption that the children of each vertex considered by the recursion are all distinct.

We may now prove our main result on dynamic regular graphs.

Proof of Theorem 6. We want to show that after n balls have been allocated to the dynamic regular graph $(G^{(1)}, \ldots, G^{(n)})$, which satisfies the ε-visibility property, the maximum load is at most $\log_2 \log n + \mathcal{O}(1/\varepsilon)$ with high probability.

Let $c > 0$ be a given constant. By the second statement of Proposition 17, with probability at least $1 - n^{-c}$, the number of cycle-producing edges in a given component of $\mathcal{C}_n$ is at most $c_2 = 2(c+1)/\varepsilon$. For a contradiction, suppose that there exists a bin, say r, which has at least $\ell + c_1 + c_2 + 1$ balls, where $c_1 = 12(c+1)$. Consider $c_2 + 1$ balls in r at height at least $\ell + c_1$. The children of r in $\mathcal{C}_n$ are the bins $r_1, r_2, \ldots, r_{c_2+1}$ (which might not be distinct), which were the alternative choice of these $c_2 + 1$ balls. Each of these children r_i has load at least $\ell + c_1$. We start the recursive construction at each child r_i of r. Assuming that this component of $\mathcal{C}_n$ contains at most c_2 cycle-producing edges, it follows that for at least one child r_i of r, the recursive procedure gives distinct children for each vertex which is a descendent of r_i. Hence we obtain a c_1-loaded tree which has 2^ℓ vertices. Substituting $\ell = \log_2 \log n$ and applying the first statement of Proposition 17, we conclude that with probability at least $1 - n^{-c}$ such a structure does not exist in $\mathcal{C}_n$. This contradiction shows that with high probability, the maximum load after n balls have been allocated is at most $\log_2 \log n + \mathcal{O}(1/\varepsilon)$. ◀

References

1 Anders Aamand, Mathias Bæk Tejs Knudsen, and Mikkel Thorup. Power of d choices with simple tabulation. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, volume 107 of *LIPIcs*, pages 5:1–5:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.5.

2 Yossi Azar, Andrei Z. Broder, Anna R. Karlin, and Eli Upfal. Balanced allocations. *SIAM J. Comput.*, 29(1):180–200, 1999. doi:10.1137/S0097539795288490.

3 Petra Berenbrink, André Brinkmann, Tom Friedetzky, and Lars Nagel. Balls into bins with related random choices. *J. Parallel Distrib. Comput.*, 72(2):246–253, 2012.

4 Paul Bogdan, Thomas Sauerwald, Alexandre Stauffer, and He Sun. Balls into bins via local search. In *Proc. 24th Symp. Discrete Algorithms (SODA)*, pages 16–34, 2013.

5 John W. Byers, Jeffrey Considine, and Michael Mitzenmacher. Geometric generalizations of the power of two choices. In *Proc. 16th Symp. Parallelism in Algorithms and Architectures (SPAA)*, pages 54–63, 2004.

6 L. Elisa Celis, Omer Reingold, Gil Segev, and Udi Wieder. Balls and bins: Smaller hash families and faster evaluation. *SIAM J. Comput.*, 42(3):1030–1050, 2013. doi:10.1137/120871626.

7 Xue Chen. Derandomized balanced allocation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2513–2526, 2019. doi:10.1137/1.9781611975482.154.

8 Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. Chernoff-hoeffding bounds for markov chains: Generalized and simplified. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, pages 124–135, 2012. doi:10.4230/LIPIcs.STACS.2012.124.

9 Andrea E. F. Clementi, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Information spreading in stationary markovian evolving graphs. *IEEE Trans. Parallel Distrib. Syst.*, 22(9):1425–1432, 2011. doi:10.1109/TPDS.2011.33.

10 Xavier Dahan. Regular graphs of large girth and arbitrary degree. *Combinatorica*, 34(4):407–426, 2014.

11 Søren Dahlgaard, Mathias Bæk Tejs Knudsen, Eva Rotenberg, and Mikkel Thorup. The power of two choices with simple tabulation. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, page 1631–1642, USA, 2016. Society for Industrial and Applied Mathematics.

12 Brighten Godfrey. Balls and bins with structure: balanced allocations on hypergraphs. In *Proc. 19th Symp. Discrete Algorithms (SODA)*, pages 511–517, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347138.

13 Krishnaram Kenthapadi and Rina Panigrahy. Balanced allocation on graphs. In *Proc. 17th Symp. Discrete Algorithms (SODA)*, pages 434–443, 2006.

14 Donald Knuth. *The Art of Computer Programming, Vol. 1: Fundamental Algorithms*. Adison-Wesley, third edition, 1997.

15 David A. Levin, Yuval Peres, and Elizabeth L. Wilmer. *Markov chains and mixing times*. American Mathematical Society, 2006. URL: http://scholar.google.com/scholar.bib?q=info:3wf9IU94tyMJ:scholar.google.com/&output=citation&hl=en&as_sdt=2000&ct=citation&cd=0.

16 Michael Mitzenmacher, Andréa W. Richa, and Ramesh Sitaraman. The power of two random choices: A survey of techniques and results. In *in Handbook of Randomized Computing*, pages 255–312. Kluwer, 2000.

17 Yuval Peres, Kunal Talwar, and Udi Wieder. Graphical balanced allocations and the $(1+\beta)$-choice process. *Random Struct. Algorithms*, DOI: 10.1002/rsa.20558, 2014.

18 Ali Pourmiri. Balanced allocation on graphs: A random walk approach. *Random Struct. Algorithms*, 55(4):980–1009, 2019. doi:10.1002/rsa.20875.

19 N. J. A. Sloane. Oeis foundation inc., the on-line encyclopedia of integer sequences, 1964. URL: http://oeis.org/A000108.

20 Berthold Vöcking. How asymmetry helps load balancing. *J. ACM*, 50(4):568–589, 2003. doi:10.1145/792538.792546.

21 Udi Wieder. Hashing, load balancing and multiple choice. *Foundations and Trends in Theoretical Computer Science*, 12(3-4):275–379, 2017. doi:10.1561/0400000070.

A Proof of Theorem 4

Proof. Let $G = ([n], E)$ denote a s-regular graph that does not contain any 4-cycle, where $s = n^{\varepsilon}$. It is worth mentioning that there are several explicit families of s-regular graphs with girth $\log_s n$ (e.g., see [10]). For each $i \in [n]$, let $N(i)$ be set of vertices adjacent to i.

Also, let $\mathcal{H} = ([n], \{N(i), i = 1, \ldots, n\})$ denote a hypergraph obtained from G. We consider the s-uniform dynamic hypergraph $(\mathcal{H}, \mathcal{H}, \ldots, \mathcal{H})$. Clearly, for every $\{i, j\} \subset [n]$ we have that

$$\texttt{vis}(i, j) \leqslant n \leqslant sn^{1-\varepsilon}$$

Therefore, the dynamic hypergraph is ε-visible. Fix an integer d such that $2 \leqslant d \leqslant s$ and d is constant. Since G does not contain any 4-cycle, we deduce that every d-subset of vertices only appears in at most one hyperedge of $\mathcal{H}$. Therefore, the probability that a d-subset is chosen by any ball is $1/(n\binom{s}{d})$. Let $D = \{i_1, i_2, \ldots, i_d\} \subset [n]$ be an arbitrary set of d vertices contained in some hyperedge of $\mathcal{H}$. Let $X(D, k)$ be an indicator random variable taking one if at least k balls choose D and zero otherwise. Then we have that

$$\mathbf{Pr}\left[X(D, k) = 1\right] = \binom{n}{k}\left(\frac{1}{n\binom{s}{d}}\right)^k$$

Also let $Y_k = \sum_D X(D, k)$ denote the number of d-subsets that are chosen by at least k balls. By linearity of expectation we have that

$$\mathbf{E}\left[Y_k\right] = \sum_D \mathbf{E}\left[X(D, k)\right] = n\binom{s}{d}\binom{n}{k}\left(\frac{1}{n\binom{s}{d}}\right)^k \geqslant n\left(\frac{s^{-d}}{k}\right)^k = n\left(\frac{n^{-d\varepsilon}}{k}\right)^k, \tag{13}$$

where the last inequality follows from $\binom{n}{k} \geqslant (\frac{n}{k})^k$ and $\binom{s}{d} < s^d$. In what follows we show that with high probability there exists k such that $Y_k \geqslant 1$. Suppose that $d\varepsilon = \Theta(1)$, then if we set $k = 1$, then there is a d-subset which is picked by at least one ball and hence $Y_1 \geqslant 1$. If $(\log\log n)/(3\log n) < d\varepsilon$ and $d\varepsilon = o(1)$, then by setting $k = 1/(6d\varepsilon)$ we have $k < (\log n)/(2\log\log n) < \log n$ and

$$\mathbf{E}\left[Y_k\right] \geqslant nk^{-k}n^{-kd\varepsilon} \geqslant n(\log n)^{-\log n/(2\log\log n)}n^{-1/6} = n^{1/3} = \omega(\log n).$$

Moreover, if $d\varepsilon \leqslant \log\log n/(3\log n)$, then by letting $k = \log n/(2\log\log n)$ we get that

$$\mathbf{E}\left[Y_k\right] \geqslant nk^{-k}n^{-kd\varepsilon} \geqslant n(\log n)^{-\log n/(2\log\log n)}n^{-1/6} = n^{1/3} = \omega(\log n).$$

Therefore, there exists $k = \min\{\Omega(1/\varepsilon), \Omega(\log n/\log\log n)\}$ so that $\mathbf{E}\left[Y_k\right] = \omega(\log n)$. As the number of balls is n, it is easy to observe that for a given k, the random variables $X(D, k)$ are negatively correlated. Application of the Chernoff bound for negatively correlated random variable implies that

$$\mathbf{Pr}\left[Y_k \leqslant \mathbf{E}\left[Y_k\right]/2\right] \leqslant \exp(-\mathbf{E}\left[Y_k\right]/8) = \exp(-\omega(\log n)).$$

It follows that there exists a d-subset D which is chosen by at least k balls and hence there is at least one bin in D whose load is at least k/d. ◀

B Proof of Proposition 8

In this section we prove Proposition 8. First we restate a useful theorem from [8].

▶ **Theorem 18** ([8, Theorem 3]). *Let M be an ergodic Markov chain with finite state space Ω and stationary distribution π. Let $T = T(\varepsilon)$ be its ε-mixing time for $\varepsilon < 1/8$. Let $(Z_1, \ldots, Z_t)$ denote a t-step random walk on M starting from an initial distribution ρ on Ω (that is, Z_1 is distributed according to ρ). For some positive constant μ and every $i \in [t]$, let $f_i : \Omega \to [0, 1]$ be a weight function at step i such that the expected weight $\mathbf{E}_\pi\left[f_i(v)\right] = \sum_{v\in\Omega}\pi(v)f_i(v)$ satisfies $\mathbf{E}_\pi\left[f_i(v)\right] = \mu$ for all i. Define the total weight of the walk $(Z_1, ..., Z_t)$ by $X = \sum_{i=1}^t f_i(Z_i)$. Write $||\rho||_\pi = \sqrt{\sum_{x\in\Omega}\rho_x^2/\pi_x}$. Then there exists some positive constant c (independent of μ and ε) such that for all $\alpha \geqslant 0$,*

1. $\mathbf{Pr}\left[X \geqslant (1+\alpha)\mu t\right] \leqslant c||\rho||_\pi \, \mathrm{e}^{-\alpha^2\mu t/72T}$ *for* $0 \leqslant \alpha \leqslant 1$.
2. $\mathbf{Pr}\left[X \geqslant (1+\alpha)\mu t\right] \leqslant c||\rho||_\pi \, \mathrm{e}^{-\alpha\mu t/72T}$ *for* $\alpha > 1$.
3. $\mathbf{Pr}\left[X \leqslant (1-\alpha)\mu t\right] \leqslant c||\rho||_\pi \, \mathrm{e}^{-\alpha^2\mu t/72T}$ *for* $0 \leqslant \alpha \leqslant 1$.

Proof of Proposition 8. Let Ω be the vertex set of the R-dimensional torus $\Gamma(n, R)$ and let a and b denote two arbitrary agents. By definition of the communication graph process, agents a and b are initially placed on two randomly chosen vertices of Γ, say u_0 and v_0. Note that u_0 and v_0 are independently chosen according to the stationary distribution π of the random walk on $\Gamma(n, R)$. Now consider the trajectory of agents a and b, which give two independent random walks $u_0, u_1, \ldots$ and $v_0, v_1, \ldots$ on $\Gamma(n, R)$. Defining $X_t = (u_t, v_t)$ for $t = 0, 1, \ldots$ gives a finite, ergodic Markov chain with stationary distribution (π, π) on $\Omega \times \Omega$. For every $t \geqslant 0$, define $f(X_t) = f(u_t, v_t)$ to equal 1 if $d(u_t, v_t) \leqslant r$, and equal 0 otherwise, where $d(\cdot, \cdot)$ is the Manhattan distance for the given grid. Let u_t^1 and v_t^1 denote the projection of the random walks u_t and v_t onto the 1-dimensional torus $\Gamma(n^{1/R}, 1)$, respectively, defined by taking the first component of each of the random walks on $\Gamma(n, R)$. Then $X_t^1 = (u_t^1, v_t^1)$ is an ergodic Markov chain on $\Gamma(n^{1/R}, 1)$, and its initial distribution is stationary. We may also define $f(u_t^1, v_t^1)$ to be 1 if $d(u_t^1, v_t^1) \leqslant r$, and 0 otherwise. By the Manhattan distance property, if $f(u_t, v_t) = 1$ then $f(u_t^1, v_t^1) = 1$. Therefore,

$$\mathtt{vis}(a, b) = \sum_{t=0}^{n} f(X_t) \leqslant \sum_{t=0}^{n} f(X_t^1).$$

Set $\delta = \min\{1/4, 1/R\}$. Let t_0 be the first time when $d(u_{t_0}^1, v_{t_0}^1) \leqslant n^\delta$. Consider a moving window W of length $2n^\delta + 1$, which contains the locations of $u_{t_0}^1$ and $v_{t_0}^1$. At time t_0, the vertices covered by W are labelled in increasing order, with the leftmost vertex labelled $-n^\delta$ and the rightmost vertex labelled $n^\delta - 1$. The window W stays at its initial location as long as no agent hits a border of W (vertices labelled $-n^\delta$ or n^δ), or the middle vertex of W (labelled 0). Let b be the first agent that hits a border or the centre of W. From this time on, b and W are coupled so that they both move and/or stay, simultaneously. (If b moves left then W also moves left, for example.) Each time the window W moves, a vertex $u \in \Gamma_1$ is no longer covered by W and a new vertex, $w \in \Gamma_1$, becomes covered by w. The new vertex w is assigned the label of vertex u. This process always labels the vertices covered by W by $\{-n^{\delta-1}, \ldots, n^\delta - 1\}$, and the movement of agent b over these labeled vertices simulates a random walk on the additive group $\mathbb{Z}_{2n^\delta+1}$. Define

$$S = \{1 \leqslant t \leqslant n \mid u_t^1 \text{ and } v_t^1 \in W\}.$$

Assume that $S \neq \emptyset$ and define the chain $Y_t = (u_t^1, v_t^1), t \in S$. Then Y_t can be considered as an ergodic Markov chain of length $|S| \leqslant n$ over $\mathbb{Z}_{2n^\delta-1}$, or equivalently, as a Markov chain on a $(2n^\delta + 1)$-cycle. By the proposition assumption we have $r = \mathcal{O}(n^{o(1)}) < n^\delta$, and so

$$\mathtt{vis}(a, b) = \sum_{t=0}^{n} f(X_t) \leqslant \sum_{t=0}^{n} f(X_t^1) \leqslant \sum_{t \in S} f(Y_t) \leqslant \sum_{t=0}^{n} f(Y_t).$$

The chain Y_t converges to stationary distribution (π, π), where π is the uniform distribution of a random walk on a $(2n^\delta + 1)$-cycle. It follows that for all $t = 0, 1, \ldots$ we have $\mathbf{E}_{(\pi,\pi)}\left[f(Y_t)\right] = \mu = \Theta(r/n^\delta)$, independently of t. It is well-known [15] that the ε-mixing time of the random walk on a $(2n^\delta + 1)$-cycle is $\mathcal{O}(n^{2\delta} \log(1/\varepsilon))$. If ρ is the initial distribution Y_0, then we have that $||\rho||_\pi \leqslant \mathcal{O}(n^\delta)$. Applying Theorem 18 implies that

$$\mathbf{Pr}\left[\sum_{t=1}^{n} f(Y_t) \geqslant \mu \cdot n\right] = \mathcal{O}(n^\delta)\mathrm{e}^{-\Theta(rn^{1-3\delta})} = n^{-\omega(1)}.$$

Therefore, with probability $1 - n^{-\omega(1)}$,

$$\texttt{vis}(a,b) \leqslant \sum_{t=0}^{n} f(Y_t) = \mathcal{O}(rn^{1-\delta}) = \mathcal{O}(n^{1-\delta+o(1)}) = \mathcal{O}(n^{1-\varepsilon}),$$

taking $\varepsilon = \delta/2$, say. Taking the union bound over all pairs of agents completes the proof. ◀

C Proof of Lemma 14

Berenbrink et al. [3] proposed an allocation algorithm $\mathcal{B}$ such that for $t = 1, 2, \ldots$, the t-th ball chooses an edge of $\mathcal{H}^{(t)} = ([n], \mathcal{E}_t), t = 1, \ldots$, uniformly at random, say H_t. The ball is then allocated to an empty vertex (bin) of H_t, with ties broken randomly. If H_t does not contain an empty bin then the process fails. The next lemma follows directly from [3, Lemmas 4, 5].

▶ **Lemma 19.** *Suppose that the dynamic s-uniform hypergraph $(\mathcal{H}^{(1)}, \ldots, \mathcal{H}^{(n)})$ satisfies the balancedness and size properties. There exists $m = \Theta(n)$ such that with probability at least $1 - n^{-2}$, algorithm $\mathcal{B}$ successfully allocates m balls and there are at least $s/2$ empty vertices in H_t for $t = 1, \ldots, m$.*

We now apply the above result to prove Lemma 14.

Proof of Lemma 14. We apply a coupling technique between the balanced allocation process on a dynamic hypergraph and $\mathcal{B}$.

Let us first consider an identical copy of the set of bins, called B. The coupled process sequentially allocates a ball to a pair of bins. In round $t = 1, \ldots, m$, the t-th ball chooses an edge of $\mathcal{H}^{(t)}$ uniformly at random, say H_t. Let H'_t be the corresponding set of bins, chosen from B. Then the first ball is allocated to a bin, say i, contained in H_t according the balanced allocation. If $i \in H'_t$ is empty then the second ball is allocated to bin $i \in H'_t$ as well. If $i \in H'_t$ is not empty then the second ball is allocated to an empty bin from H'_t, with ties are broken randomly. If there is no empty bin in H'_t then the coupling fails. Note that H_t and H'_t have the same set of bins but may have different loads. Observe that the coupled process allocates balls to bins from B according to $\mathcal{B}$. Next we show that for $t = 1, \ldots, m$,

$$\textsc{Empty}(H_t) \geqslant \textsc{Empty}(H'_t), \tag{14}$$

where $\textsc{Empty}(H)$ denotes the number of empty bins contained in H. For a contradiction, assume that there is a first time t_1 such that $\textsc{Empty}(H'_{t_1}) > \textsc{Empty}(H_{t_1})$. Then there is vertex $i \in H'_{t_1}$ which is empty, while $i \in H_{t_1}$ has a ball at height zero: this is ball t_0, say, where $1 \leqslant t_0 \leqslant t_1$. This implies that the coupled process has allocated ball t_0 to bin $i \in H_{t_1}$, but it has not allocated any ball to bin $i \in H'_{t_1}$, since i was empty until round t_1. This contradicts the definition of the coupled process. So Inequality (14) holds for $t = 1, \ldots, m$. Applying Lemma 19 yields that there exists $m = \Theta(n)$ such that for $t = 1, \ldots, m$, $\textsc{Empty}(H_t) \geqslant \textsc{Empty}(H'_t) \geqslant s/2$. ◀

D Proof of Proposition 17

In this subsection we will prove two lemmas and then combine them to establish the proposition. The lemmas and their proofs are inspired by [13, Lemma 2.1 and 2.2]. Recall that a subgraph of $\mathcal{C}_n$ is c-loaded if every vertex (bin) in the subgraph has load at least c.

▶ **Lemma 20.** *Let k be a positive integer and let $c_1 > 0$. The probability that conflict graph $\mathcal{C}_n$ contains a c_1-loaded connected component with k vertices is at most $n \cdot 8^k \cdot \left(\frac{2\mathrm{e}}{c_1}\right)^{c_1 k}$. Moreover, by setting $c_1 = 12(c+1)$, we conclude that with probability at least $1 - n^{-c}$, the conflict graph $\mathcal{C}_n$ does not contain a c_1-loaded tree with at least $\log n$ vertices.*

Proof. A connected component in $\mathcal{C}_n$ with k vertices contains a spanning tree with k vertices. By Proposition 10, there are at most 4^{k-1} ordered trees with k vertices. For every ordered tree, we can choose its root in n ways, as we have n bins (vertices). Hence there are at most $n \cdot 4^{k-1}$ rooted and ordered trees. Let us fix an arbitrary ordered tree T with a specified root. Also let $(t_1, \ldots, t_{k-1})$ denote an arbitrary sequence of rounds, where $t_i \in \{1, \ldots, n\}$ is the round when the i-th edge of the ordered tree T is chosen. Notice that in an ordered tree with specified root, the i-th edge always connects the i-th child to its parent, and the parent is already known to us. Therefore, to build the tree, the i-th edge of the tree must be chosen from edges of $G^{(t_i)}$ that are adjacent to the known parent. This implies that the algorithm chooses the i-th edge of T in round t_i with probability $\frac{\Delta_{t_i}}{n\Delta_{t_i}/2} = \frac{2}{n}$. Since balls are independent from each other, the tree T is constructed at the given times $(t_1, \ldots, t_{k-1})$ with probability

$$\left(\frac{2}{n}\right)^{k-1}. \tag{15}$$

On the other hand, ball t is allocated to a given bin with probability at most $\Delta_t/(n\Delta_t/2) = 2/n$. Therefore, the probability that T is c_1-loaded is at most

$$\binom{n}{ck}\left(\frac{2k}{n}\right)^{c_1 k} \leqslant \left(\frac{\mathrm{e}n}{c_1 k}\right)^{c_1 k}\left(\frac{2k}{n}\right)^{c_1 k} = \left(\frac{2\mathrm{e}}{c_1}\right)^{c_1 k}, \tag{16}$$

where we used the fact that $\binom{n}{c_1 k} \leqslant \left(\frac{\mathrm{e}n}{c_1 k}\right)^{c_1 k}$. Since balls are independent, one can multiply (15) by (16) and derive an upper bound for the probability that T is constructed at the given times and is c-loaded. Taking the union bound over all rooted ordered trees and time sequences gives

$$\begin{aligned} n4^{k-1} \sum_{(t_1,\ldots,t_{k-1})} \left\{\left(\frac{2}{n}\right)^{k-1}\left(\frac{2\mathrm{e}}{c_1}\right)^{c_1 k}\right\} &\leqslant n4^{k-1}n^{k-1} \cdot \left\{\left(\frac{2}{n}\right)^{k-1}\left(\frac{2\mathrm{e}}{c_1}\right)^{c_1 k}\right\} \\ &= n8^{k-1}\left(\frac{2\mathrm{e}}{c_1}\right)^{c_1 k}, \end{aligned}$$

proving the first statement of the lemma. By setting $c_1 = 12(c+1)$ and $k = \log n$ in the above formula, we infer that the probability that $\mathcal{C}_n$ contains a c_1-loaded tree with $\log n$ vertices is at most

$$n8^{k-1}\left(\frac{2\mathrm{e}}{c_1}\right)^{c_1 k} < n2^{3k}2^{-12(c+1)k} \leqslant n2^{-ck-9k} \leqslant n^{-c},$$

completing the proof. ◀

▶ **Lemma 21.** *Suppose that the conflict graph $\mathcal{C}_n$ contains a c-loaded k-vertex tree T, where $c > 4\mathrm{e}$ is any constant and k is a positive integer. Let p denotes the number of cycle-producing edges (with respect to T) which have been added between vertices in this tree during the allocation process. Then $p < 2(c+1)/\varepsilon$ with probability at least $1 - n^{-c}$.*

Proof. For a given connected component of k vertices, there are at most $\binom{k}{2}$ edges whose addition may produce a cycle. This includes edges already present in the component, as an edge with multiplicity 2 (double edge) forms a 2-cycle. Thus, the p edges can be chosen in $\binom{k}{2}^p < k^{2p}$ ways. Let $\{e_1, e_2, \ldots, e_p\}$ denote a set of p cycle-producing edges (some of these may create 2-cycles). Also let $(t_1, \ldots, t_p)$ denote a sequence of rounds, where $t_i \in \{1, \ldots, n\}$ is the round in which the t_i-th ball picks edge e_i. For each round $t = 1, 2, \ldots, n$ and $i = 1, \ldots, p$, let us define $\mathbb{I}_t(e_i)$ to equal 1 if $e_i \in E_t$, and 0 otherwise. It is easy to see that

$$\mathbf{Pr}\left[\text{ball } t \text{ picks edge } e_i \text{ of } G^{(t)}\right] = \frac{\mathbb{I}_t(e_i)}{|E_t|}.$$

Now $\texttt{vis}(e_i) = \sum_{t=1}^{n} \mathbb{I}_t(e_i)$ for $i = 1, \ldots, p$. Using this, and the fact that $|E_t| \geqslant n/2$ for each t (since $G^{(t)}$ is regular with degree at most 1), the probability that $e_1, e_2, \ldots, e_p$ are chosen is at most

$$\sum_{(t_1,\ldots,t_p)} \left\{ \prod_{i=1}^{p} \frac{\mathbb{I}_{t_i}(e_i)}{E_{t_i}} \right\} \leqslant \prod_{i=1}^{p} \left\{ \sum_{t=1}^{n} \frac{\mathbb{I}_t(e_i)}{E_t} \right\} \leqslant \prod_{i=1}^{p} \frac{4\,\texttt{vis}(e_i)}{n} \leqslant \left(\frac{4n^{1-\varepsilon}}{n}\right)^p = \left(\frac{4}{n^\varepsilon}\right)^p. \quad (17)$$

Moreover, applying Lemma 20 shows that the probability that $\mathcal{C}_n$ contains a c-loaded k-vertex tree is at most

$$n \cdot 8^k \cdot \left(\frac{2\mathrm{e}}{c}\right)^{ck} \leqslant n \cdot 2^{-k}, \quad (18)$$

as $c \geqslant 4\mathrm{e}$. So, with high probability, $\mathcal{C}_n$ does not contain any c-loaded tree with at least $(\log n)^2$ vertices. Now assume that $k < (\log n)^2$. Combining (17) and (18), and taking the union bound over all choices for a set of p edges, we find that the probability that a c-loaded k-vertex tree contains p cycle-producing edges is at most

$$k^{2p} \cdot \left(\frac{4}{n^\varepsilon}\right)^p \cdot n \cdot 2^{-k} = \left(\frac{4 \cdot k^2}{n^\varepsilon}\right)^p \cdot n \cdot 2^{-k} \leqslant n^{-\varepsilon p/2} \cdot n \cdot 2^{-k}, \quad (19)$$

where the inequality holds as $k < (\log n)^2$. Therefore the probability that $p = \lceil 2(c+1)/\varepsilon \rceil$ cycle-producing edges are present is at most n^{-c}. We conclude that $p < 2(c+1)/\varepsilon$ with probability at least $1 - n^{-c}$. ◀

Proof of Proposition 17. Combining the Lemmas 20 and 21 establishes the proposition. ◀

The GaussianSketch for Almost Relative Error Kernel Distance

Jeff M. Phillips
School of Computing, University of Utah, Salt Lake City, UT, USA
jeffp@cs.utah.edu

Wai Ming Tai
School of Computing, University of Utah, Salt Lake City, UT, USA
wmtai@cs.utah.edu

Abstract

We introduce two versions of a new sketch for approximately embedding the Gaussian kernel into Euclidean inner product space. These work by truncating infinite expansions of the Gaussian kernel, and carefully invoking the RecursiveTensorSketch [Ahle et al. SODA 2020]. After providing concentration and approximation properties of these sketches, we use them to approximate the kernel distance between points sets. These sketches yield almost $(1+\varepsilon)$-relative error, but with a small additive α term. In the first variants the dependence on $1/\alpha$ is poly-logarithmic, but has higher degree of polynomial dependence on the original dimension d. In the second variant, the dependence on $1/\alpha$ is still poly-logarithmic, but the dependence on d is linear.

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

Keywords and phrases Kernel Distance, Kernel Density Estimation, Sketching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.12

Category RANDOM

Funding Thanks to NSF CCF-1350888, ACI-1443046, CNS- 1514520, CNS-1564287, and IIS-1816149.

Acknowledgements We thank Rasmus Pagh for early conversations on this topic which helped reignite and motivate this line of thought. Part of the work was completed while the first author was visiting the Simons Institute for Theory of Computing.

1 Introduction

Kernel methods are a pillar of machine learning and general data analysis. These approaches consider classic problems such as PCA, linear regression, linear classification, k-means clustering which at their heart fit a linear subspace to a complex data set. Each of the methods can be solved by only inspecting the data via a dot product $\langle x, p\rangle$. Kernel methods, and specifically the "kernel trick," simply replaces these Euclidean dot products with a non-linear inner product operation. The two most common inner products are the polynomial kernel $K_z(x,p) = (\langle x,p\rangle + 1)^z$ and the Gaussian kernel $K(x,p) = \exp(-\|x-p\|^2)$.

The "magic" of the kernel method works mainly because of the existence of a reproducing kernel Hilbert space (RKHS) $\mathcal{H}_K$ associated with any positive definite (p.d.) kernel [43] K. It is a function space, so for any data point $x \in \mathbb{R}^d$, there is a mapping $\phi : \mathbb{R}^d \to \mathcal{H}_K$ so $\phi(x) = K(x,\cdot)$. Since $\phi(x)$ is a function with domain $\mathbb{R}^d$, and each "coordinate" of $\phi(x)$ is associated with another point $p \in \mathbb{R}^d$, there are an infinite number of "coordinates," and $\mathcal{H}_K$ can be infinite dimensional. However, since $\langle \phi(x), \phi(p)\rangle_{\mathcal{H}_K} = K(x,p)$, this embedding does not ever need to be computed, we can simply evaluate $K(x,p)$. And life was good.

However, at the dawn of the age of big data, it became necessary to try to explicitly, but approximately, compute this map ϕ. Kernel methods typically start by computing and then analyzing the $n \times n$ gram matrix K_X where $(K_X)_{i,j} = K(x_i, x_j)$ for a data sets X


© Jeff M. Phillips and Wai Ming Tai;
licensed under Creative Commons License CC-BY



of size n. As n became huge, this became untenable. In a hallmark paper, Rahimi and Recht [37] devised random Fourier features (RFFs) for p.d. kernels (with max value 1, e.g., Gaussians) that compute a random map $\tilde{\phi} : \mathbb{R}^d \to \mathbb{R}^{\tilde{D}}$ so $\langle \tilde{\phi}(x), \tilde{\phi}(p) \rangle$ is an unbiased estimate of $K(x, p)$, and with probability at least $1 - \delta$ has error $|K(x, p) - \langle \tilde{\phi}(x), \tilde{\phi}(p) \rangle| \leq \varepsilon$. For just one pair of points they require $\tilde{D} = O((1/\varepsilon^2) \log(1/\delta))$, or for all comparisons among n points $\tilde{D}_n = ((1/\varepsilon^2) \log(n/\delta))$, or for any points in a region Λ of volume $\mathsf{vol}(\Lambda) \leq V$, then $\tilde{D}_V = ((1/\varepsilon^2) \log(V/\delta))$.

However, relative-error-preserving RKHS embeddings for p.d. kernels are impossible without some restriction on the size n or domain Λ of the data. Consider n data points each far from each other so any pair $x, p \in \mathbb{R}^d$ satisfies $K(x, p) < 1/n$. In any relative-error-approximate embedding $\hat{\phi} : \mathbb{R}^d \to \mathbb{R}^{\hat{D}}$, each point must be virtually orthogonal to all other points, and hence $\Omega(n)$ dimensions are required [28].

Instead, to obtain (almost) relative-error results in big data sets, researchers have relied on other approaches such as sampling [45], exploiting structure of p.d. Gram matrices [34], devising modified RFFs for regularized kernel regression [9], or building data structures for kernel density estimate queries [12].

The kernel distance and data set embeddings. To address these difficulties, we first turn our attention from the inner product $\langle \phi(x), \phi(p) \rangle_{\mathcal{H}_K} = K(x, p)$ in the RKHS to the natural distance it implies. Before stating this distance, we generalize the inner product to point sets $P \subset \mathbb{R}^d$ (which extends naturally to probability distributions μ_P with domain $\mathbb{R}^d$). We treat P as a discrete probability distribution with uniform $1/|P|$ weight on each point. This can be represented in $\mathcal{H}_K$ as $\Phi(P) = \frac{1}{|P|} \sum_{x \in P} \phi(x)$, known as the kernel mean [33]. Indeed, for any query point $p \in \mathbb{R}^d$, the inner product $\langle \Phi(P), \phi(p) \rangle_{\mathcal{H}_K} = \frac{1}{|P|} \sum_{x \in P} K(x, p)$ is precisely the kernel density estimate at p. For two point sets $P, Q \subset \mathbb{R}^d$ we define $\kappa(P, Q) = \frac{1}{|P|} \frac{1}{|Q|} \sum_{x \in P} \sum_{y \in Q} K(p, q) = \langle \Phi(P), \Phi(Q) \rangle_{\mathcal{H}_K}$.

Now the kernel distance [36, 26] (alternatively known as the current distance [23] or maximum mean discrepancy [24, 39]) is defined

$$\mathtt{D}_K(P, Q) = \|\Phi(P) - \Phi(Q)\|_{\mathcal{H}_K} = \sqrt{\kappa(P, P) + \kappa(Q, Q) - 2\kappa(P, Q)}.$$

Under a slightly restricted class of kernels (a subset of p.d. kernels), called *characteristic* kernels [42], this distance is a metric. These include the Gaussian kernels which we focus on hereafter. This distance looks and largely acts like Euclidean distance; indeed, restricted to any finite-dimensional subspace, it is equivalent to Euclidean distance.

In data analysis and statistics, kernel mean is a compact way to represent a point set distribution. One can also use kernel distance to compare different point set as opposed to more expensive measure such as Wasserstein distance. In practice, there are various applications such as hypothesis test and geometric search (see section 4 for detail discussion) that use kernel distance as a core component. We suggest the reader refer to [38, 40] for more details on the statistical perspective of kernel distance. Therefore, making computation of the kernel distance scalable by a kernel distance embedding is of significant importance for those downstream applications. More generally, one can view oblivious kernel distance embedding as special case of oblivious subspace embedding for RKHS [32, 2], which gives a stronger guarantee than a subspace in the RKHS is preserved within relative error. However, many application of kernel distance do not require such a strong guarantee, which generally attain worse results (see below for more detail comparison).

So a natural question to ask is if this distance is preserved within relative error via some approximate lifting. Clearly RFFs guarantee additive ε-error. However, relate this problem to the Johnson-Lindenstrauss (JL) Lemma [25]: JL describe a family of random projections

from a high-dimensional space to a D'-dimensional space which preserve $(1+\varepsilon)$-relative error on Euclidean distance, again with $D' = O((1/\varepsilon^2)\log(n/\delta))$ for any $\binom{n}{2}$ pairs of distances, succeed with probability $1-\delta$, but only guarantees additive error on inner products.

Moreover, it is possible to apply the JL Lemma to create such an approximate embedding. First for any set of n points X, we can create $n \times n$ Gram matrix K_X (that is positive definite), and decompose it to $K_X = B_X B_X^T$. Then each row $(B_X)_i$ in B_X is the n-dimensional vector representation of the ith data point, and the Euclidean distance $\|(B_X)_i - (B_X)_j\|_2$ is the kernel distance between data points i and j [31, 8]. Then we can apply JL on these rows $\{(B_X)_i\}$ to obtain such an approximate embedding. However, this embedding is not oblivious to the data (necessary for many big data settings like streaming) and still requires $\Omega(n^2)$ time just to create the Gram matrix, not to mention the time for decomposition.

Another recent approach [14] analyzed RFFs for this task, and shows that these approximate embeddings do guarantee relative error on the kernel distance, but only between each pair of points $x, p \in \mathbb{R}^d$ (e.g., so $\frac{\|\hat{\phi}(x)-\hat{\phi}(p)\|}{\mathtt{D}_K(x,p)} \in (1 \pm \varepsilon)$), and as we describe next many downstream analysis tasks require the distance preserved between point *sets*. Alternatively, if we assume $\mathtt{D}_K^2(P,Q) > \alpha$, then standard RFFs can provide a relative error guarantee using $\tilde{D} = O(\frac{1}{\varepsilon^2\alpha^2}\log\frac{1}{\delta})$. However, such a large factor in α is undesirable, since typically $\alpha \ll \varepsilon$.

Our Results. We provide two sketches $G : \mathbb{R}^d \to \mathbb{R}^D$ for the Gaussian kernel, improving on work of Rahimi and Recht [37] and Avron et al. [9], which achieves almost relative error for kernel distance. Let $F(X) = \frac{1}{|X|}\sum_{x\in X} G(x)$ extend the sketch to point sets $X \subset \mathbb{R}^d$. Then we show that for two point sets $P, Q \subset \mathbb{R}^d$

$$\left|\mathtt{D}_K^2(P,Q) - \|F(P) - F(Q)\|^2\right| \leq \varepsilon \mathtt{D}_K^2(P,Q) + \alpha.$$

As we can always reduce the dimension $G : \mathbb{R}^d \to \mathbb{R}^D$ using JL to about $D = 1/\varepsilon^2$, we focus on reducing the runtime dependence, in particular the dependence on α.

In the first sketch (the GaussianSketch) to process a single point with $G(x)$ it takes $O\left(\frac{d^2}{\varepsilon^2}\log\frac{d}{\varepsilon} + ds\right)$ time, with $s = \Theta\left(\frac{\log(d\exp(dL^2)/\alpha)}{\log(\frac{1}{L^2}\log(d\exp(dL^2)/\alpha))}\right)$, where L describes the (L_∞) radius of the domain containing X. So the dependence on $1/\alpha$ is less than a single logarithmic term.

The second sketch (the GaussianSketchHD) is useful when the dimension d is potentially large (it turns out to be very similar to a recent sketch in [2], but our analysis is different). Then the runtime to compute $G(x)$ is $O\left(\frac{s^3}{\varepsilon^2}\log\frac{s}{\varepsilon} + s^2 d\right)$ where $s = \Theta\left(\frac{\log(4\exp(2R^2)/\alpha)}{\log(\frac{1}{R^2}\log(4\exp(2R^2)/\alpha))}\right)$, and R is the (L_2) domain radius. Now the dependence on $1/\alpha$ is still poly-logarithmic, but the dependence on dimension d is linear.

For example, we can set $\alpha = n^{-C_1}$, $R = C_2\sqrt{\log n}$ and $L = C_3\sqrt{\log n}$ for some absolute constant C_1, C_2, C_3. In low dimension, we have $s = \Theta(\frac{\log n}{\log d})$ and the running time is $O(\frac{d^2}{\varepsilon^2}\log\frac{d}{\varepsilon} + \frac{d\log n}{\log d})$. In high dimension, we have $s = \Theta(\log n)$ and the running time is $O\left(\frac{1}{\varepsilon^2}\log^3 n\log(\log n/\varepsilon) + d\log^2 n\right)$.

Implications. Several concrete applications work direfctly on this kernel distance between point sets. First, the kernel two-sample test [24, 33] is a non-parametric way to perform hypothesis tests between two empirical distributions; simply, the null hypothesis of them being drawn from the same distribution is rejected if the kernel distance is sufficiently large. While the sketched kernel two-sample test has proven effective under additive error [48], when the significance threshold is $\Theta(1/n)$, the RFF-based solutions require time $O(n^2)$, no better than brute force; but setting ε constant and $\alpha = 1/n$, our sketches provide near-linear or almost-linear time runtimes. Second, devising a Locality Sensitive Hash (LSH) between

point sets (or geometrically-aware LSH for probability distributions) has lacked a great general solution. Despite progress in special cases (e.g., for polygons [13], curves [18]), more general distances between geometric distributions, like Earth-Mover distance require $\Omega(\log s)$ distortion on a domain with at least s discrete points [7]. In general, an LSH requires relative error to properly provide $(1+\varepsilon)$-approximate nearest neighbor results. In Section 4 we specify how our new almost relative-error embeddings for the kernel distance provide efficient solutions for these applications.

Furthermore, this embedding can be composed with a Johnson-Lindenstrauss-type embedding [25, 3, 4, 1, 46] to create an overall oblivious embedding of dimension roughly $O(\frac{1}{\varepsilon^2}\log\frac{1}{\delta})$, that is with no dependence on $1/\alpha$ or d (or n or domain radius L or R in the *for each* setting), and roughly the same guarantees.

1.1 Comparison to Other Recent Work on Large Data and Kernels

Recent related works on kernel approximation do not provide our guarantees; we survey here work that addresses similar problems, and often require similar sets of error parameters.

Approximated KDEs. Charikar and Siminelakis [12] describe a data structure of size $n\hat{D}$ and query time $\hat{D}$, which answers $\kappa(P,t)$ queries within $(1+\varepsilon)$-relative error as long as $\kappa(P,t)>\alpha$; it requires $\hat{D}=O(\frac{1}{\varepsilon^2}\frac{1}{\sqrt{\alpha}}\log\frac{1}{\delta}e^{O(\log^{2/3} n\log\log n)})$. However, this cannot argue much about how large $\mathtt{D}_K(P,Q)$ has to be for this to achieve relative error on the kernel distance since it could be $\mathtt{D}_K(P,Q)$ is small but $\kappa(P,t)$ and $\kappa(P,P)$ are both large. Moreover, its guarantees only work for a single point set P with point queries t, not for two or more points sets P,Q, as we argue many downstream data analysis tasks require.

Approximated kernel regression. Avron et al. [9] modify the RFF embeddings using different sampling probability related to the statistical leverage in the kernel space. This approximates a λ-regularized kernel regression problem, creating a $\tilde{D}$-dimensional embedding; that is for an $n\times n$ gram matrix K_X, and a regularization parameter λ it creates a $n\times\tilde{D}$ matrix Z so $(1-\varepsilon)(K_X+\lambda I_n)\preceq ZZ^*+\lambda I_n\preceq(1+\varepsilon)(K_X+\lambda I_n)$, using $\tilde{D}=O(\frac{1}{\varepsilon^2}(L^d\log^{d/2}(n/\lambda)+\log^{2d}(n/\lambda))\log(s_\lambda(K)/\delta))$. Following our forthcoming methods for analysis, one can modify this result to $(1+\varepsilon)$-approximate the kernel distance, with an additive α term, with an embedding of dimension $D=O\left(\frac{1}{\varepsilon^2}(L^d\log^{d/2}\frac{n}{\alpha}+\log^{2d}\frac{n}{\alpha})\log\frac{n}{\delta}\right)$.

Also, Ahle et al. [2] recently showed that one can create such $\tilde{D}$-dimensional embedding where $\tilde{D}=O(\frac{1}{\varepsilon^2}(R^2+\log\frac{n}{\varepsilon\lambda})^5 s_\lambda(K_X))$ in $O(\frac{1}{\varepsilon^2}(R^2+\log\frac{n}{\varepsilon\lambda})^6 s_\lambda(K_X))$ time for each data point. Again, in our setting, one can interpret this result as $(1+\varepsilon)$-approximate the kernel distance, with an additive α term, in $O(\frac{1}{\varepsilon^2}(R^2+\log\frac{n}{\varepsilon\alpha})^6 s_\alpha(K_X))$ time.

Compared to our bounds (adapted to our problem using our techniques), these depend on n and s_λ (ours do not), the low-d one is exponential in d (ours is polynomial), and the other powers are larger.

Approximate Kernel PCA. Suppose we are given a data set $X=\{x_1,\ldots,x_n\}\subset\mathbb{R}^d$, and want to find a low rank (rank k) approximation of $X_\phi=\{\phi(x_1),\phi(x_2),\ldots,\phi(x_n)\}\in\mathcal{H}_K$. In particular, this can be described concretely in the context of the Gram matrix K_X and its decomposition $B_X B_X^T$. Given any $n\times m$ matrix M, let $[M]_k$ be its best rank-k approximation. A natural question is to find a rank-k matrix $\tilde{K}_X$ so

$$\left\|K_X-\tilde{K}_X\right\|_F^2\leq(1+\varepsilon)\left\|K_X-[K_X]_k\right\|_F^2.$$

While most previous work [19, 30, 22, 41, 44] has focused on providing absolute (or additive) error bounds. For instance, they showed roughly $\|K_X - \tilde{K}_X\|_F^2 \leq \|K_X - [K_X]_k\|_F^2 + \varepsilon n$ using e.g., Nyström sampling and RFFs. More recently, Musco and Woodruff [35] for p.d. Gram matrices K_X show how to efficiently find $\tilde{K}_X$ with relative error. This only requires $O(nk^{\omega-1} \cdot \mathsf{poly}(\log n/\varepsilon))$ inspections of entries of K_X, where $\omega < 2.373$ is the matrix multiplication exponent. This is not data oblivious, and uses properties of the p.d. matrix, so it does not provide an embedding sketch.

A closely related problem is approximate kernel PCA problem which is to find a $n \times k$ orthonormal matrix V so that

$$\|B_X - VV^T B_X\|_F^2 \leq (1+\varepsilon)\|B_X - [B_X]_k\|_F^2.$$

The RKHS basis V, provides a compact and non-linear set of attributes to describe a complex data set X, and has many uses in analyzing complex data which lacks strong linear correlations. Musco and Woodruff [34] provide an algorithm with runtime $O(\mathsf{nnz}(X)) + \tilde{O}(n^{\omega+1.5}(\frac{k}{\sigma_{k+1}\varepsilon^2})^{\omega-1.5})$; which has polynomial dependence on $1/\sigma_{k+1}$. They leave open whether this can be removed or reduced while maintaining only roughly $\mathsf{nnz}(X)$ dependence on X. The matrix V returned by their algorithm can be used to approximate the matrix K_X by writing $B_X P B_X^T$ where P is the projection onto the row span of $VV^T B_X$.

Our techniques can be combined with the a sketch for the polynomial kernel [10] to explicitly solve for V so

$$\|B_X - VV^T B_X\|_F^2 \leq (1+\varepsilon)\|B_X - [B_X]_k\|_F^2 + \alpha.$$

with similar dimensions required for approximating the kernel distance; the s parameter increases roughly by $\log n/\log\log n$. This is detailed in Appendix A. If the data size n has a known bound, then this provides an oblivious sketch for this almost relative error kernel PCA problem. Moreover, replacing the σ_{k+1} with $\varepsilon\alpha$, it almost answers the kernel PCA $\mathsf{nnz}(X)$ question of Musco and Woodruff [34] – however our algorithm does not depend on the number-of-non-zeros of X through our sketches, so we leave as an open question if our sketches $G(x)$, particular the GaussianSketchHD or similar, can be generated in time $O(\mathsf{nnz}(x)\mathsf{polylog}(1/\alpha) + n\mathsf{poly}(k, 1/\varepsilon, \log(1/\alpha))$.

2 The GaussianSketch and its Properties

In this section we describe our new sketches for approximate mapping from $\mathbb{R}^d$ to an RKHS associated with a Gaussian kernel. They are based on the RecursiveTensorSketch of Ahle et al. [2], so we first review its properties.

The RecursiveTensorSketch. We first introduce RecursiveTensorSketch hash family [2]. Given positive integers n, m and k, RecursiveTensorSketch$_{n,m,k}$ is the family of hash functions $T : \mathbb{R}^{n^k} \to \mathbb{R}^m$ as constructed in [2]. This hash family will be used to construct our main sketch and has the following guarantee [2]: suppose $u, v \in \mathbb{R}^{n^k}$ and picking $m = O(\frac{k}{\varepsilon^2})$, then the expectation $\mathsf{E}(\langle T(u), T(v)\rangle) = \langle u, v\rangle$ and the variance $\mathsf{Var}(\langle T(u), T(v)\rangle) \leq \frac{\varepsilon^2}{10}\|u\|^2\|v\|^2$. Moreover, the running time of computing $T(x)$ for any $x \in \mathbb{R}^{n^k}$ is $O(km\log m + kn)$.

The GaussianSketch. Now, we can define the hash family of the first sketch for the Gaussian kernel GaussianSketch. Given a vector $x \in \mathbb{R}^d$ and a positive integer s, we first define d vectors $y_x^{(1)} \dots, y_x^{(d)} \in \mathbb{R}^s$ such that ith coordinate of $y_x^{(j)}$ is $\exp(-x_j^2)\sqrt{\frac{2^{i-1}}{(i-1)!}}x_j^{i-1}$. Given an integer m, define GaussianSketch$_{m,s}$ to be the family of hash functions that if G is in it, then $G(x) = T(y_x^{(1)} \otimes \cdots \otimes y_x^{(d)})$ where T is randomly chosen from RecursiveTensorSketch$_{s,m,d}$.

Here, $x \otimes y$ is Kronecker product. Namely, given $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$, $x \otimes y$ is a pq dimensional vector indexed by two integers i, j where $i = 1, \dots, p$ and $j = 1, \dots, q$ such that $(x \otimes y)_{i,j} = x_i \cdot y_j$. For notational convenience, we extend Kronecker product when p and q are infinity. Namely, given $\{x_i\}_{i=1}^{\infty}$ and $\{y_j\}_{j=1}^{\infty}$ are infinite sequences, $x \otimes y$ is also an infinite sequence indexed by two positive integers i, j such that $(x \otimes y)_{i,j} = x_i \cdot y_j$. Also, denote $x^{\otimes k} = x \otimes x^{\otimes k-1}$ and $x^{\otimes 0} = 1$.

The rationale for the GaussianSketch comes from the following infinite expansion of the Gaussian kernel. Define $\bar{y}_x^{(j)}$ (for $j \in [d]$) as the infinite dimensional analog of $y_x^{(j)}$ with its ith coordinate as $\exp(-x_j^2)\sqrt{\frac{2^{i-1}}{(i-1)!}}x_j^{i-1}$.

▶ **Lemma 1.** *For $x, p \in \mathbb{R}^d$*

$$\begin{aligned}
&\exp(-\|x-p\|^2) \\
&= \sum_{j_1=0}^{\infty} \cdots \sum_{j_d=0}^{\infty} \left(\exp(-\|x\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}x_i^{j_i}\right)\right)\left(\exp(-\|p\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}p_i^{j_i}\right)\right) \\
&= \left\langle \bar{y}_x^{(1)} \otimes \cdots \otimes \bar{y}_x^{(d)}, \bar{y}_p^{(1)} \otimes \cdots \otimes \bar{y}_p^{(d)} \right\rangle.
\end{aligned}$$

Proof.

$$\begin{aligned}
&\exp(-\|x-p\|^2) \\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\exp(2\langle x, p\rangle) \\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\prod_{i=1}^{d}\exp(2x_ip_i) \\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\prod_{i=1}^{d}\left(\sum_{j=0}^{\infty}\frac{1}{j!}(2x_ip_i)^j\right) \quad \text{by Taylor expansion of } \exp(\cdot) \\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\sum_{j_1=0}^{\infty}\cdots\sum_{j_d=0}^{\infty}\left(\prod_{i=1}^{d}\frac{1}{j_i!}(2x_ip_i)^{j_i}\right) \\
&= \sum_{j_1=0}^{\infty} \cdots \sum_{j_d=0}^{\infty} \left(\exp(-\|x\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}x_i^{j_i}\right)\right)\left(\exp(-\|p\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}p_i^{j_i}\right)\right) \\
&= \left\langle \bar{y}_x^{(1)} \otimes \cdots \otimes \bar{y}_x^{(d)}, \bar{y}_p^{(1)} \otimes \cdots \otimes \bar{y}_p^{(d)} \right\rangle.
\end{aligned}$$

◀

Note that the Gaussian sketch takes as input one element of these inner products, but trimmed so that each $\bar{y}_x^{(j)}$ is trimmed to $y_x^{(j)}$ (without the $\bar{}$ marker) that only has s terms each.

The GaussianSketchHD. We can also define another hash family of sketches for the Gaussian kernel GaussianSketchHD, which works better for high dimension d, but will have worse dependence on other error and domain parameters. For $j = 1, \dots, s$, it will use T_j

as randomly chosen from RecursiveTensorSketch$_{d,m_j,j-1}$. Given a vector $x \in \mathbb{R}^d$, a positive integer s, and s positive integers $m_1, \ldots, m_s$, define GaussianSketchHD$_{m_1,\ldots,m_s,s}$ to be the family of hash functions that if G is in it, then $G(x) \in \mathbb{R}^m$ with $(m_{j-1}+1)$th coordinate to m_jth coordinate be $\sqrt{\frac{2^{j-1}}{(j-1)!}}\exp(-\|x\|^2)T_j(x^{\otimes j-1}) = T_j(z_x^{(j)}) \in \mathbb{R}^{m_j}$ where $z_x^{(j)} = \sqrt{\frac{2^{j-1}}{(j-1)!}}\exp(-\|x\|^2)x^{\otimes j-1} \in \mathbb{R}^{d^{j-1}}$ and $m = \sum_{j=1}^s m_j$. Denote z_x the $\frac{d^s-1}{d-1}$ dimensional vector where the first coordinate is $z_x^{(1)}$, the next d coordinates are $z_x^{(2)}$, the next d^2 coordinates are $z_x^{(3)}$, and so on. The GaussianSketchHD uses the following, a different infinite expansion of the Gaussian kernel (also explored by Cotter et al. [17]).

▶ **Lemma 2.** *For $x, p \in \mathbb{R}^d$,*

$$\exp(-\|x-p\|^2) = \sum_{i=0}^{\infty}\left\langle \exp(-\|x\|^2)\sqrt{\frac{2^i}{i!}}x^{\otimes i}, \exp(-\|p\|^2)\sqrt{\frac{2^i}{i!}}p^{\otimes i}\right\rangle = \sum_{i=0}^{\infty}\left\langle z_x^{(i)}, z_p^{(i)}\right\rangle$$

Proof.

$$\begin{aligned}
&\exp(-\|x-p\|^2)\\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\exp(2\langle x,p\rangle)\\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\sum_{i=0}^{\infty}\frac{1}{j!}(2\langle x,p\rangle)^j && \text{by Taylor expansion of } \exp(\cdot)\\
&= \exp(-\|x\|^2)\exp(-\|p\|^2)\sum_{i=0}^{\infty}\frac{2^j}{j!}\langle x^{\otimes j}, p^{\otimes j}\rangle\\
&= \sum_{j=0}^{\infty}\left\langle \exp(-\|x\|^2)\sqrt{\frac{2^j}{j!}}x^{\otimes j}, \exp(-\|p\|^2)\sqrt{\frac{2^i}{j!}}p^{\otimes j}\right\rangle
\end{aligned}$$

◀

2.1 Concentration Bounds for GaussianSketch and GaussianSketchHD

The sketches will inherit the concentration properties of the RecursiveTensorSketch. Similar observations were recently observed by Ahle et al. [2]. Consider a weighted set of elements $X \subset \mathbb{R}^d$ with weights α_x for $x \in X$, and we use the general concentration bounds for these under the GaussianSketch.

▶ **Lemma 3** ([2]). *Let G be a randomly chosen hash function in GaussianSketch$_{m,s}$ with $m = O\left(\frac{d}{\varepsilon^2}\right)$. Let $v = \sum_{x\in X}\alpha_x y_x^{(1)} \otimes \cdots \otimes y_x^{(d)}$, then $\mathsf{E}\left[\left\|\sum_{x\in X}\alpha_x G(x)\right\|^2\right] = \|v\|^2$ and $\mathsf{Var}\left[\left\|\sum_{x\in X}\alpha_x G(x)\right\|^2\right] \le \frac{\varepsilon^2}{10}\|v\|^4$ and hence with probability at least 9/10 we have $\left|\left\|\sum_{x\in X}\alpha_x G(x)\right\|^2 - \|v\|^2\right| \le \varepsilon\|v\|^2$.*

If G is randomly chosen from GaussianSketchHD$_{m_1,\ldots,m_s,s}$, then $G(x) = Sz_x$, where S is a $m \times \frac{d^s-1}{d-1}$ random matrix (recall $m = \sum_{j=1}^s m_j$) so, for the $(m_{i-1}+1)$th row to the m_ith row, and the $(\frac{d^{i-1}-1}{d-1}+1)$th column to the $\frac{d^i-1}{d-1}$th column forms a matrix S_i where $T_i(z_x^{(i)}) = S_i z_x^{(i)}$, and the rest of entries are zero.

▶ **Lemma 4** ([2]). *Suppose A, B has $\frac{d^s-1}{d-1}$ columns. Denote A_i and B_i be ith row of A and B respectively. By taking $m_i = O\left(\frac{i}{\varepsilon^2}\right)$, we have $\mathbf{Pr}\left[\left\|AB^T - AS^TSB^T\right\|_F^2 \le \varepsilon^2\|A\|_F^2\|B\|_F^2\right] \ge 1-\delta$.*

2.2 Truncation Bounds for GaussianSketch and GaussianSketchHD

These sketches are effective when it is useful to analyze the effect of sketching a large data set X of size n, and we desire to show the cumulative measured across all pairs of elements. For each sketch we expand these infinite sums, and determine the truncation parameter s so the sum of terms past s have a bounded effect.

In our analysis, we will use the following inequality which follows by standard calculus analysis, for any $\eta > 0$,

$$\sum_{j=s}^{\infty} \frac{\eta^j}{j!} \leq \frac{\left(\sup_{y\in[-\eta,\eta]} \exp(y)\right)\eta^s}{s!} \leq \frac{\exp(\eta)\eta^s}{s!} \tag{1}$$

The following expression also arises in our analysis.

▶ **Lemma 5.** *For $\xi, a, b > 0$, setting $s = \Theta\left(\frac{\log \frac{\xi\cdot a}{\alpha}}{\log\left(\frac{1}{b}\log\frac{\xi\cdot a}{\alpha}\right)}\right)$ then the we have $\xi \cdot a\left(\frac{b}{s}\right)^s \leq \alpha$.*

Proof. By setting $\frac{s}{b} = C\frac{\gamma}{\log\gamma}$ for some large constant C where $\gamma = \frac{1}{b}\log\frac{\xi a}{\alpha}$, we have

$$\frac{s}{b}\log\frac{s}{b} = C\frac{\gamma}{\log\gamma}\log\left(C\frac{\gamma}{\log\gamma}\right) = \gamma\cdot C\left(1 + \frac{\log C}{\log\gamma} - \frac{\log\log\gamma}{\log\gamma}\right) \geq \gamma = \frac{1}{b}\log\frac{\xi a}{\alpha}.$$

Now, if we rearrange the inequality then $\xi \cdot a\left(\frac{b}{s}\right)^s \leq \alpha$. ◀

Consider a point set $X = \{x^{(1)}, x^{(2)}, \ldots, x^{(n)}\} \subset \mathbb{R}^d$, denote K_X as the $n \times n$ matrix with $(K_X)_{i,j} = \exp(-\left\|x^{(i)} - x^{(j)}\right\|^2)$. First truncate K_X using Lemma 1 to obtain the $n \times n$ matrix $K^{\mathsf{GS}}_{X,s}$ with

$$\begin{aligned}
&(K^{\mathsf{GS}}_{X,s})_{i,j} \\
&= \sum_{j_1=0}^{s-1}\cdots\sum_{j_d=0}^{s-1}\left(\exp(-\left\|x^{(i)}\right\|^2)\left(\prod_{a=1}^{d}\sqrt{\frac{2^{j_a}}{j_a!}}(x_a^{(i)})^{j_a}\right)\right) \\
&\qquad\qquad \cdot\left(\exp(-\left\|x^{(j)}\right\|^2)\left(\prod_{a=1}^{d}\sqrt{\frac{2^{j_a}}{j_a!}}(p_a^{(j)})^{j_a}\right)\right)
\end{aligned}$$

▶ **Lemma 6.** *Suppose $X \subset \mathbb{R}^d$ so for all $x^{(i)} \in X$ has $\left\|x^{(i)}\right\|_\infty \leq L$ for some $L > 0$. Given a vector $w \in \mathbb{R}^n$ with $\left(\sum_{i=1}^n |w_i|\right)^2 \leq \xi$, we have*

$$w^T(K_X - K^{\mathsf{GS}}_{X,s})w \leq \left(\sum_{i=1}^n |w_i|\right)^2 d\exp(2dL^2)\left(\frac{2eL^2}{s}\right)^s \leq \alpha,$$

where the last $\leq \alpha$ inequality follows from setting $s = s_{L,d,\alpha} = \Theta\left(\frac{\log\frac{\xi\cdot d\exp(2dL^2)}{\alpha}}{\log\left(\frac{1}{2eL^2}\log\frac{\xi\cdot d\exp(2dL^2)}{\alpha}\right)}\right)$.

Proof. From Lemma 1, we have

$$\begin{aligned}
&(K_X - K^{\mathsf{GS}}_{X,s})_{i,j} \\
&= \sum_{\substack{j_1,\ldots,j_d \\ \text{one of } j_b \geq s}}\left(\exp(-\left\|x^{(i)}\right\|^2)\left(\prod_{a=1}^{d}\sqrt{\frac{2^{j_a}}{j_a!}}(x_a^{(i)})^{j_a}\right)\right) \\
&\qquad\qquad \cdot\left(\exp(-\left\|x^{(j)}\right\|^2)\left(\prod_{a=1}^{d}\sqrt{\frac{2^{j_a}}{j_a!}}(x_a^{(j)})^{j_a}\right)\right)
\end{aligned}$$

Then we can analyze these in aggregate with respect to a test vector z. The first line uses the fact that a matrix A (for instance with $A = K_X - K^{\mathsf{GS}}_{X,s}$) written as $\sum_j (\sum_{x_i \in X} \psi_j(x_i))(\sum_{x'_i \in X} \psi_j(x'_i))$ can be simplified $w^T A w = \sum_j (\sum_{x_i \in X} w_i \psi_j(x_i))^2$.

$$
\begin{aligned}
& w^T (K_X - K^{\mathsf{GS}}_{X,s}) w \\
&= \sum_{\substack{j_1,\ldots,j_d \\ \text{one of } j_b \geq s}} \left(\sum_{i=1}^n w_i \exp(-\left\|x^{(i)}\right\|^2) \left(\prod_{a=1}^d \sqrt{\frac{2^{j_a}}{j_a!}} (x_a^{(i)})^{j_a} \right) \right)^2 \\
&\leq \sum_{b=1}^d \sum_{\substack{j_1,\ldots,j_d \\ j_b \geq s}} \left(\sum_{i=1}^n w_i \exp(-\left\|x^{(i)}\right\|^2) \left(\prod_{a=1}^d \sqrt{\frac{2^{j_a}}{j_a!}} (x_a^{(i)})^{j_a} \right) \right)^2 && \text{by union bound} \\
&\leq \sum_{b=1}^d \sum_{\substack{j_1,\ldots,j_d \\ j_b \geq s}} \left(\sum_{i=1}^n |w_i| \left(\prod_{a=1}^d \sqrt{\frac{2^{j_a}}{j_a!}} L^{j_a} \right) \right)^2 && \text{assuming } \left\|x^{(i)}\right\|_\infty \leq L \\
&\leq \left(\sum_{i=1}^n |w_i| \right)^2 \left(\sum_{b=1}^d \sum_{\substack{j_1,\ldots,j_d \\ j_b \geq s}} \left(\prod_{a=1}^d \frac{(2L^2)^{j_a}}{j_a!} \right) \right)
\end{aligned}
$$

The term $\sum_{b=1}^d \sum_{\substack{j_1,\ldots,j_d \\ j_b \geq s}} \left(\prod_{a=1}^d \frac{(2L^2)^{j_a}}{j_a!} \right)$ can be expressed as the follows.

$$
\begin{aligned}
& \sum_{b=1}^d \sum_{\substack{j_1,\ldots,j_d \\ j_b \geq s}} \left(\prod_{a=1}^d \frac{(2L^2)^{j_a}}{j_a!} \right) \\
&= \sum_{b=1}^d \left(\sum_{j_1=0}^\infty \frac{(2L^2)^{j_1}}{j_1!} \right) \cdots \left(\sum_{j_b=s}^\infty \frac{(2L^2)^{j_b}}{j_b!} \right) \cdots \left(\sum_{j_d=0}^\infty \frac{(2L^2)^{j_d}}{j_d!} \right) \\
&= \sum_{b=1}^d \left(\prod_{\substack{a=1 \\ a \neq b}}^d \exp(2L^2) \right) \left(\sum_{j_b=s}^\infty \frac{(2L^2)^{j_b}}{j_b!} \right) \\
&\leq \sum_{b=1}^d \left(\exp((d-1)2L^2) \right) \frac{\exp(2L^2)(2L^2)^s}{s!} && \text{by (1)} \\
&= \frac{d \exp(2dL^2)(2L^2)^s}{s!} \\
&\leq d \exp(2dL^2) \left(\frac{2eL^2}{s} \right)^s && \text{by the fact } s! \geq \left(\frac{s}{e} \right)^s
\end{aligned}
$$

Thus, we have

$$
\begin{aligned}
w^T (K_X - K^{\mathsf{GS}}_{X,s}) w &\leq \left(\sum_{i=1}^n |w_i| \right)^2 \left(\sum_{b=1}^d \sum_{\substack{j_1,\ldots,j_d \\ j_b \geq s}} \left(\prod_{a=1}^d \frac{(2L^2)^{j_a}}{j_a!} \right) \right) \\
&\leq \left(\sum_{i=1}^n |w_i| \right)^2 d \exp(2dL^2) \left(\frac{2eL^2}{s} \right)^s \\
&\leq \alpha
\end{aligned}
$$

where the last inequality follows Lemma 5 using $\xi = (\sum_{i=1}^n |w_i|)^2$, $a = d \exp(2dL^2)$ and $b = 2eL^2$. ◀

Now truncate K_X based on Lemma 2 to obtain $K^{\mathsf{HD}}_{X,s}$ with

$$(K^{\mathsf{HD}}_{X,s})_{i,j} = \sum_{a=0}^{s-1} \left\langle \exp(-\left\|x^{(i)}\right\|^2)\sqrt{\frac{2^a}{a!}}(x^{(i)})^{\otimes a}, \exp(-\left\|x^{(j)}\right\|^2)\sqrt{\frac{2^a}{a!}}(x^{(j)})^{\otimes a} \right\rangle$$

▶ **Lemma 7.** *Define $\Lambda^d_R = \{x \in \mathbb{R}^d \mid \|x\|_2 \le R\}$. For a point set $X \subset \Lambda^d_R$, and a vector $w \in \mathbb{R}^n$ with $(\sum_{i=1}^n |w_i|)^2 \le \xi$, we have*

$$w^T(K_X - K^{\mathsf{HD}}_{X,s})w \le \left(\sum_{i=1}^n |w_i|\right)^2 \exp(2R^2)\left(\frac{2eR^2}{s}\right)^s \le \alpha$$

where the last $\le \alpha$ inequality follows from setting $s = s_{R,\alpha} = \Theta\left(\frac{\log \frac{\xi\cdot\exp(2R^2)}{\alpha}}{\log\left(\frac{1}{2eR^2}\log\frac{\xi\cdot\exp(2R^2)}{\alpha}\right)}\right)$.

Proof. From Lemma 2, we have

$$(K_X - K^{\mathsf{HD}}_{X,s})_{i,j} = \sum_{a=s}^{\infty} \left\langle \exp(-\left\|p^{(i)}\right\|^2)\sqrt{\frac{2^a}{a!}}(p^{(i)})^{\otimes a}, \exp(-\left\|p^{(j)}\right\|^2)\sqrt{\frac{2^a}{a!}}(p^{(j)})^{\otimes a} \right\rangle$$

Then we can analyze these in aggregate with respect to a test vector z. The first line uses the fact that a matrix A (for instance with $A = K_X - K^{\mathsf{HD}}_{X,s}$) written as $\sum_j(\sum_{x_i\in X}\psi_j(x_i))(\sum_{x'_i\in X}\psi_j(x'_i))$ can be simplified $w^TAw = \sum_j(\sum_{x_i\in X} w_i\psi_j(x_i))^2$.

$$\begin{aligned}
&w^T(K_X - K^{\mathsf{HD}}_{X,s})w \\
&= \sum_{a=s}^{\infty}\left\|\sum_{i=1}^n w_i \exp(-\left\|x^{(i)}\right\|^2)\sqrt{\frac{2^a}{a!}}(x^{(i)})^{\otimes a}\right\|^2 \\
&\le \sum_{a=s}^{\infty}\left(\sum_{i=1}^n |w_i|\left\|\exp(-\left\|x^{(i)}\right\|^2)\sqrt{\frac{2^a}{a!}}(x^{(i)})^{\otimes a}\right\|\right)^2 \\
&\le \sum_{a=s}^{\infty}\left(\sum_{i=1}^n |w_i|\sqrt{\frac{2^a}{a!}}R^a\right)^2 && \text{assuming } \left\|x^{(i)}\right\| \le R \\
&= \left(\sum_{i=1}^n |w_i|\right)^2\left(\sum_{a=s}^{\infty}\frac{(2R^2)^a}{a!}\right) \\
&\le \left(\sum_{i=1}^n |w_i|\right)^2\frac{\exp(2R^2)(2R^2)^s}{s!} && \text{by (1)} \\
&\le \left(\sum_{i=1}^n |w_i|\right)^2\exp(2R^2)\left(\frac{2eR^2}{s}\right)^s && \text{by the fact } s! \ge \left(\frac{s}{e}\right)^s \\
&\le \alpha
\end{aligned}$$

where the last inequality follows Lemma 5 using $\xi = \left(\sum_{i=1}^n |w_i|\right)^2$, $a = \exp(2R^2)$ and $b = 2eR^2$. ◀

3 Application to the Gaussian Kernel Distance

Let $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ be Gaussian kernel. Namely, for any $x, y \in \mathbb{R}^d$, $K(x,y) = \exp(-\|x-y\|^2)$. Given two point sets $P, Q \subset \mathbb{R}^d$, one can define a similarity function $\kappa(P,Q) = \frac{1}{|P|}\frac{1}{|Q|}\sum_{x\in P}\sum_{y\in Q}K(x,y)$ and a squared kernel distance

$$\mathtt{D}^2_K(P,Q) = \kappa(P,P) - 2\kappa(P,Q) + \kappa(Q,Q).$$

We make the important observation that the above formulation is equivalent to the following form which will be much simpler to fit within our framework:

$$\mathtt{D}_K^2(P,Q) = \sum_{x\in P\cup Q}\sum_{y\in P\cup Q}\beta_x\beta_y\exp(-\|x-y\|^2)$$

where β_x is $\frac{1}{|P|}$ if $x\in P$ and $-\frac{1}{|Q|}$ if $x\in Q$.

We now express $\mathtt{D}_K^2(P,Q)$ as the infinite sum using Lemma 1.

$$\begin{aligned}
&\mathtt{D}_K^2(P,Q)\\
&= \sum_{x\in P\cup Q}\sum_{y\in P\cup Q}\beta_x\beta_y\exp(-\|x-y\|^2)\\
&= \sum_{x\in P\cup Q}\sum_{y\in P\cup Q}\beta_x\beta_y\sum_{j_1=0}^{\infty}\cdots\sum_{j_d=0}^{\infty}\left(\exp(-\|x\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}x_i^{j_i}\right)\right)\\
&\qquad\qquad\cdot\left(\exp(-\|y\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}y_i^{j_i}\right)\right)\\
&= \sum_{j_1=0}^{\infty}\cdots\sum_{j_d=0}^{\infty}\left(\sum_{x\in P\cup Q}\beta_x\exp(-\|x\|^2)\left(\prod_{i=1}^{d}\sqrt{\frac{2^{j_i}}{j_i!}}x_i^{j_i}\right)\right)^2\\
&= \left\|\sum_{x\in P\cup Q}\beta_x\bar{y}_x^{(1)}\otimes\cdots\otimes\bar{y}_x^{(d)}\right\|^2,
\end{aligned}$$

where each $\bar{y}_x^{(j)}$ is an infinite dimension vector with ith coordinate $\exp(-x_j^2)\sqrt{\frac{2^{i-1}}{(i-1)!}}x_j^{i-1}$.

▶ **Theorem 8.** *For any $\varepsilon, R, \alpha > 0$, let G be randomly chosen from $\textsc{GaussianSketch}_{m,s}$ with $m = O\left(\frac{d}{\varepsilon^2}\right)$ and $s = \Theta\left(\frac{\log\frac{4d\exp(2dL^2)}{\alpha}}{\log\left(\frac{1}{2eL^2}\log\frac{4d\exp(2dL^2)}{\alpha}\right)}\right)$. Let $\Omega_L^d = \{x\in\mathbb{R}^d \mid \|x\|_\infty \le L\}$. Define a mapping function F from any $X\subset\Omega_L^d$ so $F(X) = \sum_{x\in X}G(x)$, which is a vector in $\mathbb{R}^m$. Then for any $P,Q\subset\Omega_L^d$ with probability at least $9/10$*

$$\left|\|F(P)-F(Q)\|^2 - \mathtt{D}_K^2(P,Q)\right| \le \varepsilon\mathtt{D}_K^2(P,Q) + \alpha.$$

The mapping $G:\mathbb{R}^d\to\mathbb{R}^m$ can be computed in $O\left(\frac{d^2}{\varepsilon^2}\log\frac{d}{\varepsilon} + ds\right)$ time.

Proof. To analyze the GaussianSketch, we need to account for error from two sources: from the RecursiveTensorSketch (using Lemma 3) and parameter m, and from the truncation of the Taylor expansion at s (using Lemma 6). In this case we analyze the following infinite expansion

$$\mathtt{D}_K^2(P,Q) = \left\|\sum_{x\in P\cup Q}\beta_x\bar{y}_x^{(1)}\otimes\cdots\otimes\bar{y}_x^{(d)}\right\|^2,$$

where each $\bar{y}_x^{(j)}$ is an infinite dimension vector with ith coordinate $\exp(-x_j^2)\sqrt{\frac{2^{i-1}}{(i-1)!}}x_j^{i-1}$.

Let $v = \sum_{x\in P\cup Q}\beta_x\bar{y}_x^{(1)}\otimes\cdots\otimes\bar{y}_x^{(d)}$. Then by Lemma 3 by setting $m = O(d/\varepsilon^2)$ we have with probability at least $9/10$ that

$$\left|\left\|\sum_{x\in P\cup Q}\beta_x G(x)\right\|^2 - \|v\|^2\right| \le \varepsilon\|v\|^2.$$

Next note that $(\sum_{x\in P\cup Q}|\beta_x|)^2 \leq 4 = \xi$. So by Lemma 6 the truncation by only s terms can be accounted for as

$$\mathtt{D}_K^2(P,Q) - \|v\|^2 = \beta^T\left(K_{P\cup Q} - K^{\mathsf{GS}}_{P\cup Q,s}\right)\beta \leq 4d\exp(2dL^2)\left(\frac{2eL^2}{s}\right)^2 \leq \alpha,$$

where $K_{P\cup Q}$ and $K^{\mathsf{GS}}_{P\cup Q,s}$ are defined as in Lemma 6 with $X = P\cup Q$.

Combining these together we have

$$(1-\varepsilon)\left(\mathtt{D}_K^2(P,Q)-\alpha\right) \leq (1-\varepsilon)\|v\|^2 \leq |F(P)-F(Q)| \leq (1+\varepsilon)\|v\|^2 \leq (1+\varepsilon)\mathtt{D}_K^2(P,Q).$$

and hence as desired

$$\left|\|F(P)-F(Q)\|^2 - \mathtt{D}_K^2(P,Q)\right| \leq \varepsilon\mathtt{D}_K^2(P,Q)+\alpha.$$

Recall that the running time of G for mapping a point is

$$O(dm\log m + ds) = O\left(\frac{d^2}{\varepsilon^2}\log\frac{d}{\varepsilon} + ds\right). \qquad \blacktriangleleft$$

Using the Gaussian Sketch HD for high dimensions. We first express $\exp(-\|x-y\|^2)$ as another infinite sum using Lemma 2. Starting with $\mathtt{D}_K^2(P,Q) = \sum_{x\in P\cup Q}\sum_{y\in P\cup Q}\beta_x\beta_y\exp\left(-\|x-y\|^2\right)$ where β_x is $\frac{1}{|P|}$ if $x\in P$ and $-\frac{1}{|Q|}$ if $x\in Q$, we have

$$\begin{aligned}\mathtt{D}_K^2(P,Q) &= \sum_{x\in P\cup Q}\sum_{y\in P\cup Q}\beta_x\beta_y\left\langle \exp(-\|x\|^2)\sqrt{\frac{2^i}{i!}}x^{\otimes i}, \exp(-\|y\|^2)\sqrt{\frac{2^i}{i!}}y^{\otimes i}\right\rangle \\ &= \sum_{i=0}^{\infty}\left\|\sum_{x\in P\cup Q}\beta_x\exp(-\|x\|^2)\sqrt{\frac{2^i}{i!}}x^{\otimes i}\right\|^2.\end{aligned}$$

▶ **Theorem 9.** *For any $\varepsilon, R, \alpha > 0$, let G be randomly chosen from $\textsc{GaussianSketchHD}_{m_1,\ldots,m_s,s}$ with $m_i = O\left(\frac{i}{\varepsilon^2}\right)$ and $s = \Theta\left(\frac{\log\frac{4\exp(2R^2)}{\alpha}}{\log\left(\frac{1}{2eR^2}\log\frac{4\exp(2R^2)}{\alpha}\right)}\right)$. Let $\Lambda_R^d = \{x\in\mathbb{R}^d \mid \|x\|_2 \leq R\}$. Define a mapping function F from any $X\subset\Lambda_L^d$ so $F(X) = \sum_{x\in X}G(x)$, which is a vector in $\mathbb{R}^m$ where $m = \sum_{i=1}^s m_i$. Then for any $P,Q\subset\Lambda_R^d$ with probability at least $9/10$*

$$\left|\|F(P)-F(Q)\|^2 - \mathtt{D}_K^2(P,Q)\right| \leq \varepsilon\mathtt{D}_K^2(P,Q)+\alpha.$$

The mapping $G:\mathbb{R}^d\to\mathbb{R}^m$ can be computed in $O(\frac{s^3}{\varepsilon^2}\log\frac{s}{\varepsilon} + s^2d)$ time.

Proof. Suppose $G(x)\in\mathbb{R}^m$ with $(m_{i-1}+1)$th coordinate to m_ith coordinate be $\sqrt{\frac{2^{i-1}}{(i-1)!}}\exp(-\|x\|^2)T_i(x^{\otimes i-1})$. Here, T_i is randomly chosen from $\textsc{RecursiveTensorSketch}_{d,m_i,i-1}$ for $i = 1,\ldots,s$.

We first need to invoke Lemma 4 to inherit the appropriate concentration bounds from the RecursiveTensorSketch. We use $t\times\frac{d^s-1}{d-1}$ matrices A and B as just row vectors with $t=1$, and let $A=B$. In particular, define this single row as $z = \sum_{x\in P\cup Q}\beta_x[z_x^{(1)}, z_x^{(2)},\ldots,z_x^{(s)}]$, then the conclusion of Lemma 4 is that with probability at least $1-\delta$

$$\left|\|z\|^2 - \left\|\sum_{x\in P\cup Q}\beta_x G(x)\right\|^2\right|^2 = \left\|\|z\|^2 - zS^TSz^T\right\|_F^2 \leq \varepsilon^2\|z\|^4.$$

So by Lemma 7 the truncation by only s terms can be accounted for as

$$\mathtt{D}_K^2(P,Q) - \|z\|^2 = \beta^T(K_{P\cup Q} - K^{\mathsf{HD}}_{P\cup Q,s}))\beta \leq 4d\exp(2dL^2)(2eL^2/s)^2 \leq \alpha,$$

where $K_{P\cup Q}$ and $K^{\mathsf{HD}}_{P\cup Q,s}$ are defined as in Lemma 7 with $X = P \cup Q$.

Combining these together we have

$$(1-\varepsilon)(\mathtt{D}_K^2(P,Q) - \alpha) \leq (1-\varepsilon)\|z\|^2 \leq \|F(P) - F(Q)\|^2 \leq (1+\varepsilon)\|z\|^2 \leq (1+\varepsilon)\mathtt{D}_K^2(P,Q).$$

and hence as desired

$$\left|\|F(P) - F(Q)\|^2 - \mathtt{D}_K^2(P,Q)\right| \leq \varepsilon\mathtt{D}_K^2(P,Q) + \alpha.$$

Recall that the running time of G for mapping a point is $O(\sum_{i=1}^s im_i \log m_i + id) = O(\sum_{i=1}^s \frac{i^2}{\varepsilon^2}\log\frac{i}{\varepsilon} + id) = O(\frac{s^3}{\varepsilon^2}\log\frac{s}{\varepsilon} + s^2 d)$. ◀

4 Extensions and Data Analysis Implications

There are many data analysis applications where useful sketched bounds almost immediately follow from this new embedding. Before we begin, we start by improving the dimensionality of the embedding with a simple post-processing. We can applying a Johnson-Lindenstrauss-type embedding [25, 3, 4, 1] to the m-dimensional space to obtain $O(1/\varepsilon^2)$-dimensional space that, with constant probability, preserves the distance of a pair of point sets. Furthermore, we can use median trick to boost the success probability to $1-\delta$ by running $O(\log\frac{1}{\delta})$ independent copies. For applications in kernel two-sample hypothesis testing and nearest neighbor searching, setting δ depends on the number of queries q we make, for instances the bounded number needed for k-means clustering [16], now applied to kernel k-means. These results are useful for reducing the *storage space* of data representations. Recall that the running time of JL embedding from m-dimensional space to ρ-dimensional space is $O(m\log\rho + \rho^2)$ [3, 4].

4.1 Kernel Two-Sample Test

The kernel two-sample test [24] is a "non-parametric" hypothesis test between two probability distributions represented by finite samples P and Q; let $n = |P \cup Q|$. Then this test simply calculates $\mathtt{D}_K(P,Q)$, and if the value is large enough it rejects the null hypothesis that P and Q represent the same distribution. Since its introduction a few year ago it has seen many applications and relations; see the recent 140 page survey [33]. Zhao and Deng [48] proposed to speed this test up for large sets using RFFs which improves runtime and in some cases even statistical power. While several improvements are suggested [47] including using FastFood [29], these all only provide additive ε-error.

Consider $P \sim \mu_P$ and $Q \sim \mu_Q$. If $\mu_P = \mu_Q$, then empirical distributions P, Q may have $\mathtt{D}_K(P,Q) = \Theta(1/n)$. Hence distinguishing the case of $\mu_P = \mu_Q$ from them not being equal would either require additive error $\varepsilon = \Theta(1/n)$, or relative $(1+\varepsilon)$-error with a minimum $\Theta(1/n)$ additive error. RFFs would require $\Theta(1/\varepsilon^2) = \Theta(n^2)$ dimensions, so one may just as well compute $\mathtt{D}_K(P,Q)$ exactly in $O(n^2)$ time. In our approach, we can set ε to be a constant (say $\varepsilon = 0.2$) and α to be $\Theta(1/n)$. Assuming a constant region diameter, the total running time is $O\left(\frac{n\log n}{\log\log n}\right)$ in the low dimensional case (by Theorem 8) or $O\left(\frac{n\log^2 n(\log n + d)}{\log^2\log n}\right)$ in the high dimensional case (by Theorem 9).

Another way to determine if $\mathtt{D}_K(P,Q)$ should estimate P and Q as distinct, is to run permutation tests. That is for some large number (e.g., $q = 1000$) of trials, select two sets P_j, Q_j iid from $P \cup Q$, of size $|P|$ and $|Q|$ respectively. For each generated pair we calculate

(or estimate using Theorem 8 or Theorem 9) the value of $\mathtt{D}_K(P_j, Q_j)$, and then use the 95th-percentile of these values as a threshold. Note since each P_j, Q_j is drawn from the same domain as P, Q, then the guarantees on the accuracy of the featurized estimate carries over directly even under a large q number of permutations.

4.2 LSH for Point Sets, Geometric Distributions

The new results also allow us to immediately design LSH and nearest neighbor structures for the kernel distance by relying on standard Euclidean LSH [6]. Building a search engine for low-dimensional shapes [21] has long been a goal in computational geometry and geometric modeling. A difficulty arises in that many of the best-known shape distance measures require an alignment (e.g., Frechet [20, 5] or earth movers [11]) which creates many challenges in designing LSH-type procedures. Some methods have been designed, but with limitations, e.g., on point set size for earth mover distance [7] or number of segments in curves for discrete Frechet [18]. The kernel distance provides an alternative distance for shapes, low-dimensional distributions, or curves [26]; it can encode normals or tangents as well to encode direction information of curves [23]. That is, given two shapes composed of (or approximated by) point sets P_i, P_j, the distance between the shapes is simply $\mathtt{D}_K(P_i, P_j)$.

Given a family of point sets $\mathcal{P} = \{P_1, P_2, \ldots, P_N\}$ such that each $P_i \subset \mathbb{R}^d$ has size at most n, an ε-approximate nearest neighbor of a query point set Q is a point set $\hat{P} \in \mathcal{P}$ so that $\mathtt{D}_K(\hat{P}, Q) \leq (1+\varepsilon)\min_{P_j \in \mathcal{P}} \mathtt{D}_K(P_j, Q)$. Here, we assume that $\mathtt{D}_K(P_i, P_j) \geq \alpha'$ for any $i \neq j$. For $\varepsilon \leq 1/2$, we can embed each P_j to $F(P_j) \in \mathbb{R}^D$, and then invoke the key result from Andoni and Indyk [6] for a c'-approximate nearest neighbor, so the total error factor is $c'(1+\varepsilon)$. Overall, we can retrieve a c-approximate nearest neighbor (setting $c = c'(1+\varepsilon)$) to a query $Q \subset \mathbb{R}^d$ with $O(DN^{1/c^2+o(1)})$ query time after using $O(DN^{1+1/c^2+o(1)})$ space and $O(DN^{1+1/c^2+o(1)} + N(\frac{n \log \frac{1}{\varepsilon\alpha'}}{\log\log \frac{1}{\varepsilon\alpha'}} + \frac{1}{\varepsilon^2}\log\frac{1}{\varepsilon}))$ preprocessing when d is small or $O(DN^{1+1/c^2+o(1)} + Nn(\frac{\log^2 \frac{1}{\varepsilon\alpha'}(\log \frac{1}{\varepsilon\alpha'} + d)}{\varepsilon^2 \log^2 \log \frac{1}{\varepsilon\alpha'}}))$ preprocessing when d is large, both assuming a data region with constant diameter.

References

1. Dmitris Achlioptas. Database-friendly random projections: Johnson-lindenstrauss with binary coins. *Journal of Comp. & Sys. Sci.*, 66:671–687, 2003.
2. Thomas D Ahle, Michael Kapralov, Jakob BT Knudsen, Rasmus Pagh, Ameya Velingker, David P Woodruff, and Amir Zandieh. Oblivious sketching of high-degree polynomial kernels. In *SODA*, 2020.
3. Nir Ailon and Edo Liberty. Fast dimension reduction using rademacher series on dual bch codes. *Discrete & Computational Geometry*, 42(615), 2009.
4. Nir Ailon and Edo Liberty. An almost optimal unrestricted fast johnson-lindenstrauss transform. In *SODA*, 2011.
5. Helmut Alt and Leonidas J. Guibas. Discrete geometric shapes: Matching, interpolation, and approximation: A survey, 1996. In *Handbook of Computational Geometry*.
6. Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, 2006.
7. Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In *SODA*, 2008.
8. N. Aronszajn. Theory of reproducing kernels. *Trans. AMS*, 68:337–404, 1950. URL: `http://www.jstor.org/stable/1990404`.

9 Haim Avron, Michael Kapralov, Cameron Musco, Chistopher Musco, Ameya Velingker, and Amir Zandier. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *ICML*, 2017.

10 Haim Avron, Huy L. Nguyen, and David P. Woodruff. Subspace embeddings for the polynomial kernel. In *NIPS*, 2014.

11 Khanh Do Ba, Huy L. Nguyen, Huy N. Nguyen, and Ronnit Rubinfeld. Sublinear time algorithms for earth mover's distance. *Theory Comput Syst*, 48:428–442, 2011.

12 Moses Charikar and Paris Siminelakis. Hashing-based-estimators for kernel density in high dimensions. In *FOCS*, 2017.

13 Edgar Chaźez, Ana C. Chávez Cáliz, and Jorge L. López-López. Affine invariants of generalized polygons and matching under affine transformations. *Computational Geometry: Theory and Applications*, 58:60–69, 2017.

14 Di Chen and Jeff M. Phillips. Relative error embeddings for the gaussian kernel distance. In *Algorithmic Learning Theory*, 2017.

15 Kenneth L Clarkson and David P Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 205–214. ACM, 2009.

16 Michael B. Cohen, Sam Elder, Cameron Musco, Christopher Musco, and Mădălina Persu. Dimensionality reduction for k-means clustering and low rank approximation. In *STOC*, 2015.

17 Andrew Cotter, Joseph Keshet, and Nathan Srebro. Explicit approximations of the gaussian kernel. *arXiv preprint*, 2011. `arXiv:1109.4603`.

18 Anne Driemel and Francesco Silvestri. Locality-sensitive hashing of curves. In *33rd International Symposium on Computational Geometry*, 2017.

19 Petros Drineas and Michael W Mahoney. On the nyström method for approximating a gram matrix for improved kernel-based learning. *The Journal of Machine Learning Research*, 6:2153–2175, 2005.

20 Thomas Eiter and Heikki Mannila. Computing discrete Frechet distance. Technical report, Christian Doppler Laboratory for Expert Systems, 1994.

21 Thomas Funkhouser, Patrick Min, Michael Kazhdan, Joyce Chen, Alex Halderman, David Dobkin, and David Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 22:83–105, 2003.

22 Mina Ghashami, Daniel Perry, and Jeff M. Phillips. Streaming kernel principal component analysis. In *AIStats*, 2016.

23 Joan Glaunès and Sarang Joshi. Template estimation form unlabeled point set data and surfaces for computational anatomy. In *Math. Found. Comp. Anatomy*, 2006.

24 Arthur Gretton, Marsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alex J. Smola. A kernel two-sample test. *Journal of Machine Learning Research*, 13:723–773, 2012.

25 William B. Johnson and Joram Lindenstrauss. Extensions of Lipschitz maps into a Hilbert space. *Contemporary Mathematics*, 26:189–206, 1984.

26 Sarang Joshi, Raj Varma Kommaraju, Jeff M. Phillips, and Suresh Venkatasubramanian. Comparing distributions and shapes using the kurrent distance. In *Proceedings 27th Annual Symposium on Computational Geometry*, 2011. `arXiv:1001.0591`.

27 Ravi Kannan, Santosh Vempala, and David Woodruff. Principal component analysis and higher correlations for distributed data. In *Conference on Learning Theory*, pages 1040–1057, 2014.

28 Kasper Green Larsen and Jelani Nelson. Optimality of the johnson-lindenstrauss lemma. In *FOCS*, 2017.

29 Quoc Le, Tamás Sarlós, and Alex Smola. Fastfood – approximating kernel expansions in loglinear time. In *ICML*, 2013.

30 David Lopez-Paz, Suvrit Sra, Alex Smola, Zoubin Ghahramani, and Bernhard Schölkopf. Randomized nonlinear component analysis. *ICML*, 2014.

31 J. Mercer. Functions of positive and negative type, and their connection with the theory of integral equations. *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, 209:441–458, 1909.

32 Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *arXiv preprint*, 2016. `arXiv:1605.09522`.

33 Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel mean embedding of distributions: A review and beyond. *Foundations and Trends in Machine Learning*, 10:1–141, 2017.

34 Cameron Musco and David Woodruff. Is input sparsity time possible for kernel low-rank approximation? In *NeurIPS*, 2017.

35 Cameron Musco and David P. Woodruff. Sublinear time low-rank approximation of positive semidefinite matrices. In *FOCS*, 2017.

36 Jeff M. Phillips and Suresh Venkatasubramanian. A gentle introduction to the kernel distance. Technical report, arXiv, 2011. `arXiv:1103.1625`.

37 Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2007.

38 Dino Sejdinovic, Bharath Sriperumbudur, Arthur Gretton, and Kenji Fukumizu. Equivalence of distance-based and rkhs-based statistics in hypothesis testing. *The Annals of Statistics*, pages 2263–2291, 2013.

39 Alex J. Smola, Arthur Gretton, Le Song, and Bernhard Schölkopf. A Hilbert space embedding for distributions. In *ICALT*, 2007.

40 Bharath Sriperumbudur et al. On the optimal estimation of probability measures in weak and strong topologies. *Bernoulli*, 22(3):1839–1893, 2016.

41 Bharath Sriperumbudur and Nicholas Sterge. Approximate kernel pca using random features: Computational vs. statistical trade-off. Technical report, arXiv, 2018. `arXiv:1706.06296`.

42 Bharath K. Sriperumbudur, Kenji Fukumizu, and Gert R. G. Lanckriet. Universality, characteristic kernels and rkhs embedding of measures. *JMLR*, pages 2389–2410, 2011.

43 Bharath K. Sriperumbudur, Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf, and Gert R. G. Lanckriet. Hilbert space embeddings and metrics on probability measures. *Journal of Machine Learning Research*, 11:1517–1561, 2010.

44 Enayat Ullah, Poorya Mianjy, Teodor V. Marinov, and Raman Arora. Streaming kernel pca with $\tilde{o}(\sqrt{n})$ random features. In *NeruIPS*, 2018.

45 Shusen Wang, Alex Gittens, and Michael W. Mahoney. Scalable kernel k-means clustering with nystrom approximation: Relative-error bounds. *JMLR*, arXiv, (to appear). `arXiv:1706.02803`.

46 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10:1–157, 2014.

47 Wojciech Zaremba, Arthur Gretton, and Matthew Blaschko. B-tests: Low variance kernel two-sample tests. In *NIPS*, 2013.

48 Ji Zhao and Deyu Meng. Fastmmd: Ensemble of circular discrepancy for efficient two-sample test. *Neural Computation*, 27:1354–1372, 2015.

A Gaussian Kernel PCA

Let k be a positive integer and $\varepsilon > 0$. Avron et al. [10] provide the following algorithm. Suppose S and T are randomly chosen from $\text{RecursiveTensorSketch}_{s,m,d}$ and $\text{RecursiveTensorSketch}_{s,r,d}$ respectively where $m = \Theta(d(k^2 + \frac{k}{\varepsilon}))$ and $r = \Theta(\frac{dm^2}{\varepsilon^2})$. Given n vectors $v^{(1)}, \ldots, v^{(n)} \in \mathbb{R}^{s^d}$, compute $n \times m$ matrix M with ith row as $S(v^{(i)})$ and $n \times r$ matrix N that ith row as $T(v^{(i)})$. Let U be the orthonormal basis for column space of M and W be $m \times k$ matrix containing top k left singular vector of $U^T N$. Finally, return $V = UW$. This algorithm has the following guarantee.

▶ **Lemma 10** ([10] with straightforward modification). *Given a n-by-s^d matrix A, a positive integer k and $\varepsilon > 0$. The above algorithm that has rows of A as input returns a matrix V such that*

$$\left\|A - VV^T A\right\|_F^2 \leq (1+\varepsilon)\left\|A - [A]_k\right\|_F^2$$

where $[A]_k$ is the best rank-k approximation of A.

Now, we can directly modify the above algorithm into our context for rank-k Gaussian low-rank approximation. Given a point set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and a positive integer s. Suppose G and H are randomly chosen from $\textsc{GaussianSketch}_{m,s}$ and $\textsc{GaussianSketch}_{r,s}$ respectively. Recall that $m = \Theta(d(k^2 + \frac{k}{\varepsilon}))$ and $r = \Theta(\frac{dm^2}{\varepsilon^2})$. Compute the $n \times m$ matrix M with ith row as $G(x_i)$ and $n \times r$ matrix N with ith row as $H(x_i)$. Let U be the orthonormal basis for column space of M and W be $m \times k$ matrix containing top k left singular vector of $U^T N$. Finally, return $V = UW$.

▶ **Theorem 11.** *Let $\varepsilon, L, \alpha > 0$ and $s = \Theta\left(\frac{\log \frac{4n^2 d \exp(2dL^2)}{\alpha}}{\log\left(\frac{1}{2eL^2}\log \frac{4n^2 d\exp(2dL^2)}{\alpha}\right)}\right)$. For $\Omega_L^d = \{x \in \mathbb{R}^d \mid \|x\|_\infty \leq L\}$ and $X \subset \Omega_L^d$, and let A_X be a pd matrix with elements $(A_X)_{i,j} = K(x_i, x_j) = \exp(-\|x_i - x_j\|^2)$ for $x_i, x_j \in X$ and factorization $A_X = B_X B_X^T$. Then with constant probability*

$$\left\|B_X - VV^T B_X\right\|_F^2 \leq (1+\varepsilon)\left\|B_X - [B_X]_k\right\|_F^2 + \alpha.$$

The runtime to compute V is $O\left(nds + n\frac{d^4(k^2+\frac{k}{\varepsilon})^3}{\varepsilon^2}\right)$.

Proof. Let $v_x^{(i)}$ be a vector in $\mathbb{R}^s$ with jth coordinate to be $\exp(-x_i^2)\sqrt{\frac{2^{j-1}}{(j-1)!}}x_i^{j-1}$ for any $x \in \mathbb{R}^d$.

By Lemma 10, taking A_s as an $n \times s^d$ matrix with ith row as $v_{x_i}^{(1)} \otimes \cdots \otimes v_{x_i}^{(d)}$. We have

$$\left\|A_s - VV^T A_s\right\|_F^2 \leq (1+\varepsilon)\left\|A_s - [A_s]_k\right\|_F^2$$

From Lemma 6, $v^T(B_X B_X^T - A_s A_s^T)v \leq \left(\sum_{i=1}^n |v_i|\right)^2 d\exp(2dL^2)\left(\frac{2eL^2}{s}\right)^s \leq \alpha/n$. To see this expression is at most α/n, first observe that columns of V are orthonormal, and therefore, the norm of each row of $I - VV^T$ is at most 2. Hence, $\left(\sum_{i=1}^n |v_i|\right)^2 \leq 4n$. Then the choice of s and Lemma 5 with $\xi = 4n^2$, $a = d\exp(2dL^2)$ and $b = 2eL^2$ complete this derivation.

We now have

$$\begin{aligned}
\left\|B_X - VV^T B_X\right\|_F^2 &= \mathsf{Tr}((I - VV^T)B_X B_X^T (I - VV^T)^T) \\
&\leq \left\|A_s - VV^T A_s\right\|_F^2 + \mathsf{Tr}((I - VV^T)(B_X B_X^T - A_s A_s^T)(I - VV^T)^T) \\
&\leq \left\|A_s - VV^T A_s\right\|_F^2 + \alpha
\end{aligned}$$

On the other hand, by Lemma 1, $B_X B_X^T - A_s A_s^T$ is still positive definite. Therefore,

$$\begin{aligned}
&\|A_s - [A_s]_k\|_F^2 \\
&= \left\|A_s - UU^T A_s\right\|_F^2 \quad \text{where } U \text{ is the matrix of top-}k \text{ left singular vectors of } A_s \\
&\leq \left\|A_s - U'U'^T A_s\right\|_F^2 \quad \text{where } U \text{ is the matrix of top-}k \text{ left singular vectors of } B_X \\
&= \|B_X - [B_X]_k\|_F^2 - \mathsf{Tr}((I - U'U'^T)(B_X B_X^T - A_s A_s^T)(I - U'U'^T)) \\
&\leq \|B_X - [B_X]_k\|_F^2 \quad \text{recall that } B_X B_X^T - A_s A_s^T \text{ is positive definite}
\end{aligned}$$

We can plug in everything.

$$\begin{aligned}\left\|B_X - VV^T B_X\right\|_F^2 &\leq \left\|A_s - VV^T A_s\right\|_F^2 + \alpha \\ &\leq \|A_s - [A_s]_k\|_F^2 + \alpha \\ &\leq \|B_X - [B_X]_k\|_F^2 + \alpha.\end{aligned}$$

To see the running time, it takes $O(d(s + m\log m))$ to compute $G(\cdot)$ and $O(d(s + r\log r))$ time to compute $H(\cdot)$, and hence n times as much to compute matrices M and N. We can compute the basis U of M in $O(nm^2)$ time, and the projection $U^T N$ in $O(nrm)$ time. The basis W takes $O(rm^2)$ time, and the final low rank basis $V = UW$ takes $O(nmk)$ time. Thus the total runtime is $O(nd(s + m\log m + r\log r) + nm^2 + nrm + rm^2 + nmk) = O(nd(s + rm))$ using that $r > m^2 > k^4$ that $m > \log r$, and assuming $n > r$. Now using $m = O(d(k^2 + k/\varepsilon))$ and $r = O(dm^2/\varepsilon^2) = O(d^3(k^4 + k^2/\varepsilon^2)/\varepsilon^2)$ and we have a total time of $O\left(nds + n\frac{d^4(k^2+\frac{k}{\varepsilon})^3}{\varepsilon^2}\right)$. ◀

Gaussian Low Rank Approximation with Gaussian Sketch HD in High Dimensions. Now, we can also modify the above algorithm into our context for rank-k Gaussian low-rank approximation in another way. Given a point set $X = \{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ and a positive integer s. Suppose G and H are randomly chosen from $\textsc{GaussianSketchHD}_{m_1,\ldots,m_s,s}$ and $\textsc{GaussianSketchHD}_{r_1,\ldots,r_s,s}$ respectively. Here, $m_i = \Theta(i(k^2 + \frac{k}{\varepsilon}))$ and $r_i = \Theta(\frac{im^2}{\varepsilon^2})$ where $m = \sum_{i=1}^{s} m_i$. Compute the $n \times m$ matrix M with ith row as $G(x_i)$ and $n \times r$ matrix N with ith row as $H(x_i)$. Let U be the orthonormal basis for column space of M and W be $m \times k$ matrix containing top k left singular vector of $U^T N$. Finally, return $V = UW$.

Note that a hash function in GaussianSketchHD is not directly applying a hash function in RecursiveTensorSketch. Therefore, Lemma 10 cannot be directly applied. However, we can still exploit the structure of it in order to prove the same lemma.

As Avron et al. [10] suggest, it is generally possible by combining Lemma 4 and arguments in [10, 15, 27]. We have the following lemma. Here, denote A_s is a $n \times \frac{d^s-1}{d-1}$ matrix that ith row as z_{x_i} for given point set $X = \{x_1, x_2 \ldots, x_n\} \subset \mathbb{R}^d$.

▶ **Lemma 12.** *Given a point set $X \subset \mathbb{R}^d$, a positive integer k and $\varepsilon > 0$. The above algorithm returns a matrix V such that*

$$\left\|A_s - VV^T A_s\right\|_F^2 \leq (1+\varepsilon)\|A_s - [A_s]_k\|_F^2$$

where $[A]_k$ is the best rank-k approximation of A.

Before getting into Lemma 12, the following lemma from [10] which is implied by Lemma 4 would be helpful.

▶ **Lemma 13** ([10] implied by Lemma 4 with straightforward modification)**.** *For any positive integer k', given any $\frac{d^s-1}{d-1} \times k'$ matrix B with orthonormal columns, we have $\left\|B^T S^T SB - I\right\|_2 \leq \varepsilon$. Here, S is randomly chosen from $\textsc{GaussianSketchHD}_{n_1,\ldots,n_s,s}$ where $n_i = \frac{ik'^2}{\varepsilon^2}$.*

Proof. (of Lemma 12)

In the proof of Theorem 3.1 from [15], the only properties of S used are

- Given any $\frac{d^s-1}{d-1} \times k$ matrix B with orthonormal columns, we have $\left\|B^T S^T SB - I\right\|_2 \leq \varepsilon_0$ for some constant $\varepsilon_0 > 0$
- For any two matrices A, B with $\frac{d^s-1}{d-1}$ columns, $\left\|AB^T - AS^T SB^T\right\|_F \leq \sqrt{\frac{\varepsilon}{k}}\|A\|_F\|B\|_F$

The first property can be shown by Lemma 13 since we pick $m_i = \Omega(ik^2)$ and the second property can be shown by Lemma 4 since we pick $m_i = \Omega(\frac{ik}{\varepsilon})$. Also, Theorem 3.1 of [15] implies Lemma 4.2 of [15] which means there is a matrix Z such that $\|UZ - A_s\|_F \leq (1+\varepsilon)\|A_s - [A_s]_k\|_F$ in our context. Combining Lemma 4.3 of [15], we have

$$\left\|U[U^T A_s] - A_s\right\|_F \leq (1+\varepsilon)\|A_s - [A_s]_k\|_F \tag{2}$$

Now, Lemma 13 implies Lemma 2.1 from [27] and further implies

$$\left\|WW^T U^T A_s - A_s\right\|_f \leq (1+\varepsilon)\|A_s - [A_s]_k\|_F \tag{3}$$

by setting k' in Lemma 13 be m and picking $r_i = \Theta(\frac{3^i m^2}{\varepsilon^2})$. Using equation (2) and (3) in the proof of Theorem 1.1 from [27], we have our conclusion $\left\|A_s - UWW^T U^T A_s\right\|_F^2 = \left\|A_s - VV^T A_s\right\|_F^2 \leq (1+\varepsilon)\|A_s - [A_s]_k\|_F^2$. ◀

▶ **Theorem 14.** *Let $\varepsilon, R, \alpha > 0$ and $s = \Theta\left(\frac{\log \frac{4n^2\exp(2R^2)}{\alpha}}{\log\left(\frac{1}{2eR^2}\log\frac{4n^2\exp(2R^2)}{\alpha}\right)}\right)$. For $\Lambda_R^d = \{x \in \mathbb{R}^d \mid \|x\|_2 \leq R\}$ and $X \subset \Lambda_R^d$, and let A_X be a pd matrix with elements $(A_X)_{i,j} = K(x_i, x_j) = \exp(-\|x_i - x_j\|^2)$ for $x_i, x_j \in X$ and factorization $A_X = B_X B_X^T$. Then with constant probability*

$$\left\|B_X - VV^T B_X\right\|_F^2 \leq (1+\varepsilon)\|B_X - [B_X]_k\|_F^2 + \alpha.$$

The runtime to compute V is $O(nds^2 + n\frac{3^{4s}(k^2+\frac{k}{\varepsilon})^3}{\varepsilon^2})$.

Proof. By Lemma 12, we have

$$\left\|A_s - VV^T A_s\right\|_F^2 \leq (1+\varepsilon)\|A_s - [A_s]_k\|_F^2.$$

From Lemma 7, $v^T(B_X B_X^T - A_s A_s^T)v \leq \left(\sum_{i=1}^n |v_i|\right)^2 \exp(2R^2)\left(\frac{2eR^2}{s}\right)^s \leq \alpha/n$ with our setting of s as long as $\left(\sum_{i=1}^n |v_i|\right)^2 \leq 4n$. Indeed the columns of V are orthonormal, so the norm of each row of $I - VV^T$ is at most 2, and thus $\left(\sum_{i=1}^n |v_i|\right)^2 \leq 4n$.

We now have

$$\begin{aligned}
\left\|B_X - VV^T B_X\right\|_F^2 &= \mathsf{Tr}((I - VV^T)B_X B_X^T (I - VV^T)^T) \\
&\leq \left\|A_s - VV^T A_s\right\|_F^2 + \mathsf{Tr}((I - VV^T)(B_X B_X^T - A_s A_s^T)(I - VV^T)^T) \\
&\leq \left\|A_s - VV^T A_s\right\|_F^2 + n \cdot (\alpha/n)
\end{aligned}$$

Also by Lemma 2, $B_X B_X^T - A_s A_s^T$ is still positive definite. Therefore,

$$\begin{aligned}
&\|A_s - [A_s]_k\|_F^2 \\
&= \left\|A_s - UU^T A_s\right\|_F^2 \qquad \text{where } U \text{ is the matrix of top-}k \text{ left singular vectors of } A_s \\
&\leq \left\|A_s - U'U'^T A_s\right\|_F^2 \qquad \text{where } U' \text{ is the matrix of top-}k \text{ left singular vectors of } B_X \\
&= \|B_X - [B_X]_k\|_F^2 - \mathsf{Tr}((I - U'U'^T)(B_X B_X^T - A_s A_s^T)(I - U'U'^T)) \\
&\leq \|B_X - [B_X]_k\|_F^2 \qquad \text{recall that } B_X B_X^T - A_s A_s^T \text{ is positive definite}
\end{aligned}$$

We can plug in everything.

$$\begin{aligned}
\left\|B_X - VV^T B_X\right\|_F^2 &\leq \left\|A_s - VV^T A_s\right\|_F^2 + \alpha \\
&\leq \|A_s - [A_s]_k\|_F^2 + \alpha \\
&\leq \|B_X - [B_X]_k\|_F^2 + \alpha
\end{aligned}$$

To see the running time, it takes $O(\sum_{i=1}^{s} i(d + m_i \log m_i))$ to compute $G(\cdot)$ and $O(\sum_{i=1}^{s} i(d + r_i \log r_i))$ time to compute $H(\cdot)$. Using that $r_i > m_i^2 > k^4$ and $m_i > 1/\varepsilon$ then it takes less time to compute $H(\cdot)$ than $G(\cdot)$, and this runtime is $O(ds^2 + s^2 r_s \log r_s) = O(ds^2 + s^2 r \log r)$ since the r_i values are exponentially increasing in i, and so $r_s = O(r)$ for $r = \sum_{i=1}^{s} r_i$. The time to compute M and N is n time longer.

We can compute the basis U of M in $O(nm^2)$ time, and the projection $U^T N$ in $O(nrm)$ time – this step is the post-sketch bottlneck. The basis W takes $O(rm^2)$ time, and the final low rank basis $V = UW$ takes $O(nmk)$ time. Thus the total runtime is $O(n(ds^2 + s^2 r \log r) + nm^2 + nrm + rm^2 + nmk) = O(n(ds^2 + rm))$ using that $r > m^2 > k^4$ that $m > s^2 \log r$, and assuming $n > r$. Now using $m = O(s^2(k^2 + k/\varepsilon))$ and $r = O(sm^2/\varepsilon^2) = O(s^3(k^4 + k^2/\varepsilon^2)/\varepsilon^2)$ and we have a total time of $O(nds^2 + ns^4(k^2 + \frac{k}{\varepsilon})^3/\varepsilon^2)$. ◀

A Fast Binary Splitting Approach to Non-Adaptive Group Testing

Eric Price
Department of Computer Science, University of Texas at Austin, TX, USA
https://www.cs.utexas.edu/~ecprice/
ecprice@cs.utexas.edu

Jonathan Scarlett
Department of Computer Science & Department of Mathematics,
National University of Singapore, Singapore
https://www.comp.nus.edu.sg/~scarlett/
scarlett@comp.nus.edu.sg

Abstract

In this paper, we consider the problem of noiseless non-adaptive group testing under the for-each recovery guarantee, also known as probabilistic group testing. In the case of n items and k defectives, we provide an algorithm attaining high-probability recovery with $O(k \log n)$ scaling in both the number of tests and runtime, improving on the best known $O(k^2 \log k \cdot \log n)$ runtime previously available for any algorithm that only uses $O(k \log n)$ tests. Our algorithm bears resemblance to Hwang's adaptive generalized binary splitting algorithm (Hwang, 1972); we recursively work with groups of items of geometrically vanishing sizes, while maintaining a list of "possibly defective" groups and circumventing the need for adaptivity. While the most basic form of our algorithm requires $\Omega(n)$ storage, we also provide a low-storage variant based on hashing, with similar recovery guarantees.

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

Keywords and phrases Group testing, sparsity, sublinear-time decoding, binary splitting

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.13

Category RANDOM

Funding *Eric Price*: Supported in part by NSF Award CCF-1751040 (CAREER).
Jonathan Scarlett: Supported by an NUS Early Career Research Award.

1 Introduction

Group testing is a classical statistical problem with a combinatorial flavor [4, 19, 20], and has recently regained significant attention following new applications [14, 18, 25], connections with compressive sensing [26, 27], and most recently, utility in testing for COVID-19 [28, 42, 45]. The goal is to identify a defective (or infected) subset of items (or individuals) based on a number of suitably-designed tests, with the binary test outcomes indicating whether or not the test includes at least one defective item.

In both classical studies and recent works, considerable effort has been put into the development of group testing algorithms achieving a given recovery criterion with a near-optimal number of tests [1, 3, 5, 11, 15, 16, 20, 33, 35, 36, 39], and many solutions are known with decoding time linear or slower in the number of items. In contrast, works seeking to further reduce the decoding time have only arisen more recently [9, 10, 12, 30, 31, 34, 37]; see Section 1.2 for an overview.

In this paper, we present a non-adaptive group testing algorithm that guarantees high-probability recovery of the defective set with $O(k \log n)$ scaling in both the number of tests and the decoding time in the case of n items and k defectives. By comparison, the best


© Eric Price and Jonathan Scarlett;
licensed under Creative Commons License CC-BY



previous decoding time alongside $O(k \log n)$ tests was $O(k^2 \log k \cdot \log n)$ [9], with faster algorithms incurring suboptimal $O(k \log k \cdot \log n)$ scaling in the number of tests [10,34]. By a standard information-theoretic lower bound [4, Sec. 1.4], the $O(k \log n)$ scaling in the number of tests is optimal whenever $k \leq n^{1-\Omega(1)}$.

Before outlining the related work and contributions in more detail, we formally introduce the problem.

1.1 Problem Setup

We consider a set of n items indexed by $\{1, \dots, n\}$, and let $S \subset \{1, \dots, n\}$ be the set of defective items. The number of defectives is denoted by $k = |S|$; this is typically much smaller than n, so we assume that $k \leq \frac{n}{2}$. For clarity of exposition, we will present our algorithm and analysis for the case that k is known, but these will trivially extend to the case that only an upper bound $\overline{k} \geq k$ is known, and $\overline{k}$ replaces k in the number of tests and decoding time.

A sequence of t tests $X^{(1)}, \dots, X^{(t)}$ is performed, with $X^{(i)} \in \{0,1\}^n$ indicating which items are in the i-th test, and the resulting outcomes are $Y^{(i)} = \bigvee_{j \in S} X_j^{(i)}$ (i.e., 1 if there is any defective item in the test, 0 otherwise). We focus on the non-adaptive setting, in which all tests $X^{(1)}, \dots, X^{(t)}$ must be designed prior to observing any outcomes.

We consider a for-each style recovery guarantee, in which the goal is to develop a randomized algorithm that, for any fixed defective set S of cardinality k, produces an estimate $\widehat{S}$ satisfying $\mathbb{P}[\widehat{S} \neq S] \leq \delta$ for some small $\delta > 0$ (typically $\delta \to 0$ as $n \to \infty$). In our algorithm, only the tests $\{X^{(i)}\}_{i=1}^t$ will be randomized, and the decoding algorithm producing $\widehat{S}$ from the test outcomes will be deterministic.

1.2 Related Work

The existing works on group testing vary according to the following defining features [4, 20]:

- **For-each vs. for-all guarantees.** In *combinatorial (for-all) group testing*, one seeks to construct a test design that guarantees the recovery of *all* defective sets up to a certain size. In contrast, in *probabilistic (for-each) group testing*, the test design may be randomized, and the algorithm is allowed some non-zero probability of error.
- **Adaptive vs. non-adaptive.** In the *adaptive* setting, each test may be designed based on all previous outcomes, whereas in the *non-adaptive setting*, all tests must be chosen prior to observing any outcomes. The non-adaptive setting is often preferable in practice, as it permits the tests to be performed in parallel.
- **Noiseless vs. noisy.** In the *noiseless* setting, the test outcomes are perfectly reliable, whereas in *noisy settings*, some tests may be flipped according to some probabilistic or adversarial noise model.

As outlined in the previous subsection, our focus is on the for-each guarantee, non-adaptive testing, and noiseless tests, but for comparison, we also discuss other variants in this section.

By a simple counting argument, even in the least stringent scenario of noiseless adaptive testing and the for-each guarantee, $\Omega\big(k \log \frac{n}{k}\big)$ tests are required for reliable recovery [6, 36]. In the noiseless adaptive setting, the classical generalized binary splitting algorithm of Hwang [29] matches this bound with sharp constant factors, and attains the stronger for-all guarantee. More recently, $O(k \log n)$ adaptive tests and decoding time were shown to suffice under the for-each guarantee in the presence of random noise [10].

Table 1 Overview of existing noiseless non-adaptive group testing results with $\text{poly}(k \log n)$ decoding time under the for-each guarantee, with n items and k defectives.

Reference	Number of tests	Decoding time	Construction
SAFFRON [34]	$O(k \cdot \log k \cdot \log n)$	$O(k \log k)$[1]	Randomized
GROTESQUE [10]	$O(k \cdot \log k \cdot \log n)$	$O(k \cdot \log k \cdot \log n)$	Randomized
Inan *et al.* [30]	$O\big(k \cdot \log n \cdot \log \frac{\log n}{\log k}\big)$	$O\big(k^3 \cdot \log n \cdot \log \frac{\log n}{\log k}\big)$	Explicit
BMC [9]	$O(k \log n)$	$O(k^2 \cdot \log k \cdot \log n)$	Randomized
This Paper	$O(k \log n)$	$O(k \log n)$	Randomized

In the non-adaptive setting with the for-each guarantee, a recent of result of [2] shows that n tests are required to attain asymptotically vanishing error probability as $n \to \infty$ with $k = \lfloor \beta n \rfloor$, for arbitrarily small $\beta > 0$. Thus, $O(k \log \frac{n}{k})$ scaling (with a universal implied constant) is impossible; see also [16]. On the other hand, $O(k \log n)$ tests do suffice, and this scaling is equivalent to $O(k \log \frac{n}{k})$ in the regime $k \leq n^{1-\Omega(1)}$. Early works achieved this for the scaling regime $k = O(1)$ [5, 35, 36], and a variety of more recent works considered more general k [1, 5, 15, 16, 33, 35, 36, 39], particularly under "unstructured" random test designs such as i.i.d. Bernoulli [3, 5, 39] and near-constant tests-per-item [15, 33]. In fact, following the recent work of Coja-Oghlan et al. [16], the precise constants for non-adaptive group testing have been nailed down in the regime $k \leq n^{1-\Omega(1)}$, with the algorithm for the upper bound requiring $\Omega(n)$ decoding time.

Since non-adaptive algorithms with $\text{poly}(k \log n)$ decoding time under the for-each guarantee are particularly relevant to the present paper, we provide a summary of the existing works in Table 1. We separate our discussion according to whether or not each algorithm attains $O(k \log n)$ scaling in the number of tests, as we believe this to be a particularly important desiderata in practice when tests are expensive:

- SAFFRON and GROTESQUE both use $O(k \log k \cdot \log n)$ tests, which is suboptimal by a $\log k$ factor. In particular, SAFFRON's decoding time is as low as $O(k \log k)$ in the word-RAM model.[1] While storage requirements were not a significant point of focus in the existing works, we also note (for later comparison) that SAFFRON only requires $O(k \log k \cdot \log n)$ bits of storage; see Appendix A for details.
- In the regime $k = \Theta(n^\alpha)$ with fixed $\alpha \in (0, 1)$, Inan et al. [30] attain $O(k^3 \log n)$ decoding time with $O(k \log n)$ tests, whereas the number of tests becomes suboptimal in sparser regimes such as $k = \text{poly}(\log n)$. The best previous decoding time for an algorithm that only uses $O(k \log n)$ tests is $O(k^2 \log k \cdot \log n)$, via bit-mixing coding (BMC) [9]. To our knowledge, all other existing algorithms succeeding with $O(k \log n)$ tests incur $\Omega(n)$ decoding time [1, 5, 15, 16, 33, 35, 36, 39].

Our main theoretical contribution is to provide an algorithm that attains $O(k \log n)$ scaling in both the number of tests and decoding time, as well as using distinct algorithmic ideas from the existing works (see Section 2.1 for discussion). In particular, among the algorithms using $O(k \log n)$ tests, ours reduces the dependence on k in the decoding time from quadratic to linear.

[1] The decoding time of SAFFRON is typically stated as $O(k \log k \cdot \log n)$, but can be $O(k \log k)$ in the word-RAM model of computation, where the test outcomes are stored as $O(k \log k)$ words of length $\log_2 n$ each; see Appendix A for details.

It is worth noting that the works [9, 10, 30, 34] also extend their guarantees to noisy settings, and the approach in [30] has the additional advantage of using an *explicit* test design, i.e., the test vectors $X^{(1)}, \dots, X^{(t)}$ can be constructed deterministically in time polynomial in n and t. Although noisy and/or de-randomized variants of our algorithm may be possible, this is deferred to future work.[2]

In addition, [34] demonstrated that $O\big(k \log n \cdot \log \frac{1}{\epsilon}\big)$ tests suffice for SAFFRON in the case that one is only required to identify a fraction $1 - \epsilon$ of the defectives, as opposed to the entire defective set. Thus, $O(k \log n)$ scaling is maintained for approximate recovery with constant $\epsilon > 0$, but not for the more common requirement of exact recovery.

While we have focused our discussion on the for-each setting, the first $\text{poly}(k \log n)$-time non-adaptive group testing algorithms were for the more stringent for-all setting [12, 31, 37]. The strength of the for-all guarantee comes at the expense of requiring significantly more tests, e.g., see [23] for an $\Omega\big(\min\big\{k^2 \frac{\log n}{\log k}, n\big\}\big)$ lower bound. An $O(k^2 \log n)$ upper bound on the number of tests was originally attained with $\Omega(n)$ decoding time, [22, 38], and more recently with $\text{poly}(k \log n)$ decoding time [31, 37]. The earlier work of [12] attained $\text{poly}(k \log n)$ decoding time under a list decoding recovery criterion, and also allowed for adversarial noise in the test outcomes.

1.3 Summary of Results

Before summarizing our main results, we briefly highlight that our algorithmic techniques are distinct from the existing works summarized above (see Section 2.1 for further details), and are more closely related to the adaptive binary splitting approach of Hwang [29]. We first test large groups of items together, placing each group into a single randomly-chosen test among a sequence of $O(k)$ tests. We then "split" these groups into smaller sub-groups, while using the $O(k)$ test outcomes to eliminate those known to be non-defective. This process is repeated (with the elimination step ensuring that the number of groups under consideration does not grow too large) until a superset of S is found. This superset is shown to be of size $O(k)$ with high probability, and S is deduced from this superset via a final sequence of $O(k \log k)$ tests (similar ideas to this final step have appeared in works such as [12, 37]). Despite the sequential nature of this procedure, the tests can be performed non-adaptively.

Our first main result is informally stated as follows; see Theorem 1 for a formal statement.

▶ **Main Result 1** (Algorithmic Guarantees – Informal Version). *There exists a non-adaptive group testing algorithm that, for any constant $c > 0$, succeeds with probability $1 - O(k^{-c})$ using $O(k \log n)$ tests and $O(k \log n)$ decoding time, and requires $O(n \log^2 k)$ bits of storage.*

Main Result 1 uses $\Omega(n)$ storage to record the random choices of tests the items are included in. For our second main result, we consider a low-storage variant in which the tests are instead allocated using hash functions. In this case, the precise decoding time and storage guarantees depend on the properties of the hash family used. The following theorem informally states the guarantee resulting from $O(\log k)$-wise independent hashing using the classical hash family of Wegman and Carter [44]; see Theorem 11 for a formal and more general statement.

▶ **Main Result 2** (Algorithmic Guarantees with Reduced Storage – Informal Version). *There exists a non-adaptive group testing algorithm that, for any constant $c > 0$, succeeds with probability $1 - O\big(\frac{\log n}{k^c}\big)$ using $O(k \log n)$ tests and $O(k \log k \cdot \log n)$ decoding time, and requires $O(k \log n + \log k \cdot \log^2 n)$ bits of storage.*

[2] Our analysis can easily be adapted to handle false positive tests, but handling false negative tests appears to require more significant changes and a slightly increased decoding time.

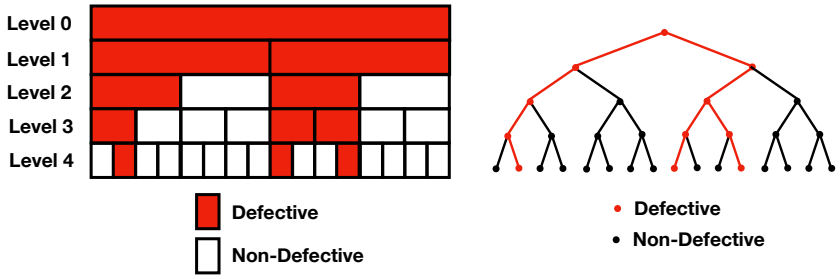


Figure 1 Example structure of groups of items that are tested together when $n = 16$ and $k = 3$ (Left), and the corresponding tree representation (Right). Leaves correspond to single items, and nodes at the higher levels correspond to groups of items.

2 Non-Adaptive Binary Splitting Algorithm

In this section, we introduce our algorithm, and state and prove our main result giving guarantees on its correctness, number of tests, decoding time, and storage. For simplicity of notation, we assume throughout the analysis that k and n are powers of two. Our algorithm only requires an upper bound on the number of defectives, and hence, any other value of k can simply be rounded up to a power of two. In addition, the total number of items n can be increased to a power of two by adding "dummy" non-defective items.

2.1 Description of the Algorithm

We propose an algorithm that works with a tree structure, as illustrated in Figure 1. On the left of the figure, we have $1 + \log_2 n$ levels, with the top level containing all items, the second level containing two groups with half the items each, and so on, with each level "splitting" the previous levels' groups in half until the final level containing individual items. The ordering of items is inconsequential for our analysis and results provided that the two children of a given node can be identified in constant time, so for convenience we assume the natural order.[3] For $\ell = 0, \dots, \log_2 n$, the j-th group at the ℓ-th level is denoted by $G_j^{(\ell)} \subseteq \{1, \dots, n\}$, and we observe that for fixed ℓ, there are 2^ℓ groups $G_1^{(\ell)}, \dots, G_{2^\ell}^{(\ell)}$ of cardinality $\frac{n}{2^\ell}$ each.

We consider the binary tree representation in Figure 1 (Right), in which each node corresponds to a group of items (e.g., the root corresponds to the set of all items, and the leaves correspond to individual items). The levels of the tree are indexed by $\ell = 0, \dots, \log_2 n$. Our algorithm works down the tree one level at a time, keeping a list of *possibly defective* (PD) nodes, and performing tests to obtain such a list at the next level, while ideally keeping the size of the list small (e.g., $O(k)$). When we perform tests at a given level, we treat each node as a "super-item"; including a node in a test amounts to including all of the items in the corresponding group $G_j^{(\ell)}$. While this may appear to naturally lead to an adaptive algorithm, we can in fact perform all of the tests non-adaptively.

If we were to test nodes at the root or the early levels, they would almost all return positive, and hence not convey significant information. We therefore let the algorithm start at the level $\ell_{\min} = \log_2 k$, and we consider the following test design:

[3] Alternatively, one may use a dyadic splitting approach: Assign each item a unique $(\log_2 n)$-bit string, and first split according to the first bit, then the second bit, and so on.

- For each $\ell = \log_2 k, \dots, \log_2 n - 1$, form a sequence of Ck tests (for some $C > 0$ to be selected later), and for each $j = 1, \dots, 2^\ell$, choose a single such test uniformly at random, and place all items from $G_j^{(\ell)}$ into that test.
- For the final level $\ell = \log_2 n$, each $G_j^{(\ell)} = \{j\}$ is a singleton. In this case, we form $C' \log k$ sequences of $2k$ tests (for some $C' > 0$). For each item, and each of the $C' \log k$ sequences of tests, we place the item in one of the $2k$ corresponding tests, chosen uniformly at random.

This creates a total of $t = Ck \log_2 \frac{n}{k} + 2C'k \log k = O(k \log n)$ tests.

Upon observing the t non-adaptive test outcomes, the decoder forms an estimate $\widehat{S}$ of the defective set via the following procedure:

- Initialize $\mathcal{G}^{(\ell_{\min})} = \{G_j^{(\ell_{\min})}\}_{j=1}^k$, where $\ell_{\min} = \log_2 k$;
- Iterate the following for $\ell = \log_2 k, \dots, \log_2 n - 1$:
  - For each group $G \in \mathcal{G}^{(\ell)}$, check whether the single test corresponding to that group is positive or negative. If positive, then add both children of G (see Figure 1) to $\mathcal{G}^{(\ell+1)}$.
- Let the estimate $\widehat{S}$ of the defective set be the (singleton) elements of $\mathcal{G}^{(\log_2 n)}$ that are not included in any negative test among the $2C'k \log k$ tests at the final level.

Comparisons with existing methods

Our algorithm is distinct from existing sublinear-time non-adaptive group testing algorithms, which are predominantly based on code concatenation [31,37] or related methods that encode item indices' binary representations directly into the test matrix [8, 10, 34]. In fact, in the context of group testing, our approach is perhaps most reminiscent of the *adaptive* algorithm of Hwang [29] (see also [4, 20]), which identifies one defective at a time using binary splitting, removing each defective from consideration after it is identified. By keeping track of the multiple defective nodes simultaneously and allowing a small number of false positives up to the final level, we are able to exploit similar ideas without requiring adaptivity.

Beyond group testing and binary splitting, this recursive tree-structured approach is also reminiscent of ideas used in sketching algorithms, such as dyadic tree recovery for count-min sketch [17]. Our particular approach – maintaining a superset at each level, and relying on a lack of false negatives – is most similar to the pyramidal reconstruction method for compressive sensing under the earth mover's distance [32].

A nearly identical algorithm to ours was given independently in the concurrent work of [13], with similar theoretical guarantees. In addition, [13] observes that since the total number of non-discarded nodes in the algorithm is $O\big(k \log \frac{n}{k}\big)$ with overwhelming probability (see Lemma 6 below), one can take a union bound over the $\binom{n}{k}$ possible defective sets to attain *list recovery* in the combinatorial (for-all) setting. This idea allows the authors of [13] to attain a variety of other results, including in the related problems of heavy hitters and compressive sensing. In the context of probabilistic (for-each) group testing, some slight advantages of our analysis include (i) proving that the first batch of tests leads to $O(k)$-size list recovery rather than $O\big(k \log \frac{n}{k}\big)$-size, thus circumventing the need for a third batch of tests, and (ii) only requiring $O(\log k)$-wise independence in our low-storage variant based on hashing (see Section 3), rather than $O\big(k \log \frac{n}{k}\big)$-wise independence (albeit at the expense of a higher error probability).

2.2 Algorithmic Guarantees

In the following, we provide the formal version of Main Result 1. Here and subsequently, we assume a word-RAM model of computation; for instance, with n items and t tests, it takes constant time to read a single integer in $\{1, \dots, n\}$ from memory, perform arithmetic operations on such integers, fetch a single test outcome indexed by $\{1, \dots, t\}$, and so on.

▶ **Theorem 1** (Algorithmic Guarantees). *Let S be a fixed (defective) subset of $\{1,\dots,n\}$ of cardinality k. For any constant $c>0$, there exist choices of C and C' such that the group testing algorithm described in Section 2.1 satisfies the following with probability $1-O(k^{-c})$:*

- *The returned estimate $\widehat{S}$ equals S;*
- *The algorithm runs in time $O(k\log n)$;*
- *The algorithm uses $O(n(\log k)^2)$ bits of storage.*

In addition, the number of tests scales as $O(k\log n)$.

We note that the success probability approaches one as $n\to\infty$ in any asymptotic regime satisfying $k=\omega(1)$, but not in the regime $k=O(1)$; the same is true in the existing works listed in Table 1, with the exception of [9]. In addition, it is straightforward to improve the success probability to $1-e^{-\Omega(k)}-O(n^{-c})$ by using $O(\log n)$ (rather than $O(\log k)$) sequences of $2k$ tests at the final level.

Theorem 1 is proved in the remainder of the section. We emphasize that our focus is on scaling laws, and we make no significant effort to optimize the underlying constant factors.

2.3 Analysis

Throughout the analysis, the defective set S will be fixed but otherwise arbitrary, and we will condition on fixed placements of the defective items into tests (and hence, fixed test outcomes). The test placements of the non-defective items are independent of those of the defectives, and our analysis will hold regardless of which particular tests the defectives were placed in. We use $\mathbb{P}[\cdot\,|\,\mathcal{T}_S]$ to represent this conditioning, and we refer to $\mathcal{T}_S$ as the defective test placements. In addition, for the tree illustrated in Figure 1, we refer to nodes containing defective items as defective nodes, to all other nodes as non-defective nodes, and to the set of defective nodes as the defective (sub-)tree.[4]

We begin with the following simple lemma.

▶ **Lemma 2** (Probabilities of Non-Defectives Being in Positive Tests). *Under the test design described in Section 2.1, the following holds at any given level $\ell=\log_2 k,\dots,\log_2 n-1$: Conditioned on any defective test placements $\mathcal{T}_S$, any given non-defective node at level ℓ has probability at most $\frac{1}{C}$ of being placed in a positive test.*

Proof. Since there are k defective items, at most k nodes at a given level can be defective, and since each node is placed in a single test, at most k tests out of the Ck tests at the given level can be positive. Since the test placements are independent and uniform, it follows that for any non-defective node, the probability of being in a positive test is at most $\frac{1}{C}$. ◀

In view of this lemma, when starting at any non-defective child of any given defective node (or alternatively, starting at a non-defective node at level $\ell_{\min}$), we can view any further branches down the non-defective sub-tree as "continuing" (i.e., the two children remain "possibly defective") with probability at most $\frac{1}{C}$. Hence, we have the following.

▶ **Lemma 3** (Probability of Reaching a Non-Defective Node). *Under the setup of Lemma 2, any given non-defective node having Δ non-defective ancestor nodes is reached (i.e., all of its ancestor nodes are placed in positive tests, so the node is considered possibly defective) with probability at most $\left(\frac{1}{C}\right)^{\Delta}$.*

We will use the preceding lemmas to control the following two quantities:

[4] Since we start at level $\ell_{\min}=\log_2 k$, this could more precisely be considered as a forest.

- N_{total}, the total number of non-defective nodes that are reached in the sense of Lemma 3;
- N_{leaf}, the number of non-defective leaf nodes that are reached, and thus are still considered possibly defective by the final level.

It will be useful to upper bound N_{total} for the purpose of controlling the overall decoding time, and to upper bound N_{leaf} for the purpose of controlling the number of items considered at the final level.

2.3.1 Bounding N_{total} and N_{leaf} on average

We first present two lemmas bounding the averages of N_{total} and N_{leaf}, and then establish high-probability bounds.

▶ **Lemma 4** (Bounding N_{total} on Average). *For any $C \geq 4$ and any defective test placements $\mathcal{T}_S$, we have*

$$\mathbb{E}[N_{\text{total}} \,|\, \mathcal{T}_S] \leq 6k \log_2 \frac{n}{k}. \tag{1}$$

Proof. Since the tree is binary, each defective node can have at most 2^i non-defective descendants appearing exactly i levels further down the tree; such descendants correspond to $\Delta = i - 1$ in Lemma 3. Since there are at most $k \log_2 \frac{n}{k}$ defective nodes, it follows that there are at most $\big(k \log_2 \frac{n}{k}\big) 2^{\Delta+1}$ non-defective nodes with Δ non-defective ancestors and at least one defective ancestor. Similarly, there are at most k non-defective nodes at level $\ell_{\min} = \log_2 k$, each of which has at most 2^Δ non-defective nodes appearing Δ levels later.

Since Lemma 3 demonstrates a probability $\big(\frac{1}{C}\big)^\Delta$ of being reached for a given number Δ of non-defective ancestors, it follows that

$$\mathbb{E}[N_{\text{total}} \,|\, \mathcal{T}_S] \leq \Big(k \log_2 \frac{n}{k}\Big) \sum_{\Delta=0}^{\log_2 \frac{n}{k} - 1} 2^{\Delta+1} \Big(\frac{1}{C}\Big)^\Delta + k \sum_{\Delta=0}^{\log_2 \frac{n}{k}} 2^\Delta \Big(\frac{1}{C}\Big)^\Delta. \tag{2}$$

Hence, the assumption $C \geq 4$ gives

$$\mathbb{E}[N_{\text{total}} \,|\, \mathcal{T}_S] \leq \Big(2k \log_2 \frac{n}{k}\Big) \sum_{\Delta=0}^{\log_2 \frac{n}{k} - 1} 2^{-\Delta} + k \sum_{\Delta=0}^{\log_2 \frac{n}{k}} 2^{-\Delta} \leq 6k \log_2 \frac{n}{k}, \tag{3}$$

where we used $\sum_{\Delta=0}^{\infty} 2^{-\Delta} = 2$ and $\log_2 \frac{n}{k} \geq 1$ (for $k \leq \frac{n}{2}$). ◀

▶ **Lemma 5** (Bounding N_{leaf} on Average). *For any $C \geq 4$ and any defective test placements $\mathcal{T}_S$, we have*

$$\mathbb{E}[N_{\text{leaf}} \,|\, \mathcal{T}_S] \leq 6k. \tag{4}$$

Proof. Again using the fact that each defective node can have at most 2^i descendants appearing i levels further down the tree, we find that there are at most $k 2^{\Delta+1}$ leaf nodes with Δ non-defective ancestors and at least one defective ancestor. In addition, a leaf having only non-defective ancestors corresponds to $\Delta = \log_2 \frac{n}{k}$ in Lemma 3 (since $\ell_{\min} = \log_2 k$), and to handle this case, we trivially upper bound he number of leaves by n. Hence, for $C \geq 4$, we have similarly to (3) that

$$\mathbb{E}[N_{\text{leaf}} \,|\, \mathcal{T}_S] \leq 2k \sum_{\Delta=0}^{\log_2 \frac{n}{k} - 1} 2^\Delta \Big(\frac{1}{C}\Big)^\Delta + n \Big(\frac{1}{C}\Big)^{\log_2 \frac{n}{k}} \leq 6k. \tag{5}$$

◀

2.3.2 Bounding N_{total} with high probability

In this subsection, we prove the following.

▶ **Lemma 6** (High-Probability Bound on N_{total})**.** *For any $C \geq 16$, conditioned on any defective test placements $\mathcal{T}_S$, we have $N_{\text{total}} = O\big(k \log \frac{n}{k}\big)$ with probability $1 - e^{-\Omega(k \log \frac{n}{k})}$.*

To prove this result, we make use of Lemmas 2 and 4, along with the following auxiliary result written in generic notation.

▶ **Lemma 7** (Sub-Exponential Behavior in a Branching Process)**.** *Consider an infinite-depth binary tree in which each node is independently assigned a value $\{0, 1\}$ with probability q of being 1, and let N be the number of nodes whose ancestors are all marked as 1. Then, when $q \leq \frac{1}{16}$, we have all $n > 0$ that $\mathbb{P}[N = n] \leq 2^{-(n-1)}$.*

Proof. We make use of branching process theory [24, Ch. XII], in particular noting that N equals the *total progeny* [21] when the initial population size is 1 (i.e., the root) and the distribution of the number of children per node is given by

$$P_X(0) = 1 - q, \quad P_X(2) = q, \quad P_X(x) = 0, \forall x \notin \{0, 2\}. \tag{6}$$

The results we use from branching process theory are expressed in terms of the generative function $M_X(s) = \mathbb{E}[s^X]$, which is given by

$$M_X(s) = (1 - q) + qs^2. \tag{7}$$

It is evident in our case that $N < \infty$ with probability one provided that $q < \frac{1}{2}$, and one way to formally prove this is to utilize the general result that the *extinction probability* $\mathbb{P}[N < \infty]$ equals one provided that the smallest root of $s = M_X(s)$ is $s = 1$ [24, Sec. XII.4].

We utilize an exact expression for the distribution of N for general branching processes [21] (specialized to the case that the initial population size is one):

$$\mathbb{P}[N = n] = \frac{1}{n} m_{n-1}^{(n)}, \tag{8}$$

where $m_{n-1}^{(n)}$ is computed according to the expansion

$$\big(M_X(s)\big)^n = \sum_{j=0}^{\infty} m_j^{(n)} s^j. \tag{9}$$

Substituting our expression (7) for $M_X(s)$ on the left-hand side, we find that $(M_X(s))^n = \big((1 - q) + qs^2\big)^n$, which equals $\sum_{i=0}^{n} \binom{n}{i} (1 - q)^{n-i} (qs^2)^i$ by a binomial expansion. Hence, we obtain $m_{n-1}^{(n)} = 0$ for even-valued n, and for odd-valued n we substitute $i = \frac{n-1}{2}$ to obtain

$$m_{n-1}^{(n)} = \binom{n}{\frac{n-1}{2}} (1 - q)^{\frac{n+1}{2}} q^{\frac{n-1}{2}} \tag{10}$$

$$\leq 2(2\sqrt{q})^{n-1}, \tag{11}$$

where we applied $\binom{n}{\frac{n-1}{2}} \leq 2^n = 2 \cdot 2^{n-1}$ and $(1 - q)^{\frac{n+1}{2}} \leq 1$.

Substituting (11) into (8), we obtain

$$\mathbb{P}[N = n] \leq \frac{2}{n} \big(\sqrt{4q}\big)^{n-1} \mathbb{1}\{n \text{ is odd}\}. \tag{12}$$

This implies $\mathbb{P}[N = n] \leq 2^{-(n-1)}$ when $q \leq \frac{1}{16}$ and $n \geq 2$, and the same trivially holds when $n = 1$. ◀

The condition $\mathbb{P}[N = n] \le 2^{-(n-1)}$ in Lemma 7 implies that N is a sub-exponential random variable, and the same trivially follows for a branching process that only runs up to a certain depth (number of generations). In the group testing setup of Lemma 6, we are adding together $O\big(k \log \frac{n}{k}\big)$ independent copies of such random variables (each corresponding to a different non-defective sub-tree) to get N_{total}. As a result, we can apply a standard concentration bound for sums of independent sub-exponential random variables [43, Prop. 5.16] to obtain

$$\mathbb{P}[N_{\text{total}} \ge \mathbb{E}[N_{\text{total}} \,|\, \mathcal{T}_S] + t \,|\, \mathcal{T}_S] \le e^{-\Omega(\min\{t^2/(k \log_2 \frac{n}{k}), t\})}, \tag{13}$$

from which Lemma 6 follows by setting $t = \Theta\big(k \log_2 \frac{n}{k}\big)$ and using $\mathbb{E}[N_{\text{total}} \,|\, \mathcal{T}_S] = O\big(k \log_2 \frac{n}{k}\big)$ (see Lemma 4). The condition $C \ge 16$ coincides with $q \le \frac{1}{16}$ in Lemma 7.

2.3.3 Bounding N_{leaf} with high probability

In this subsection, we prove the following.

▶ **Lemma 8** (High-Probability Bound on N_{leaf})**.** *For any $C \ge 12$, conditioned on any defective test placements $\mathcal{T}_S$, we have $N_{\text{leaf}} = O(k)$ with probability $1 - e^{-\Omega(k)}$.*

We again use Lemma 2, along with the following auxiliary results written in generic notation.

▶ **Lemma 9** (Tail Bound on Binary Tree Paths)**.** *Consider a binary tree of height h in which each node is independently assigned a value $\{0, 1\}$ with probability q of being 1, and let N_h be the number of leaf nodes that have a path of 1's back to the root (including both endpoints). Then $\mathbb{P}[N_h \ge t] \le 4^{-(h+t)}$ for any integer $t \ge 1$.*

Proof. We use a proof by induction. The base case is $h = 1$, in which case we have $\mathbb{P}[N_h > 2] = 0$, $\mathbb{P}[N_h = 2] = \frac{1}{q^3}$, and $\mathbb{P}[N_h \ge 1] \le \frac{2}{q^2}$ by the union bound. These bounds satisfy the claim of the lemma due to the assumption $q \le \frac{1}{12}$.

Now fix $h \ge 2$ and suppose that the claim is true for height $h - 1$. Let L and R be the number of leaf nodes reached in the left and right sub-trees of the root, and observe that

$$\mathbb{P}[N_h \ge t] \le \sum_{j=0}^{t} \mathbb{P}[L \ge j \cap R \ge t - j] \tag{14}$$

$$= \mathbb{P}[L \ge t] + \mathbb{P}[R \ge t] + \sum_{j=1}^{t-1} \mathbb{P}[L \ge j \cap R \ge t - j]. \tag{15}$$

Applying the induction hypothesis, we find that the first two terms are at most $q4^{-(h-1+t)} = 4q \cdot 4^{-(h+t)}$, and for $1 \le j \le t-1$, we have $\mathbb{P}[L \ge j \cap R \ge t-j] \le q \cdot 4^{-(h-1+j)} \cdot 4^{-(h-1+t-j)} = 16q \cdot 4^{-(2h+t)}$, where the multiplications by q correspond to the root node being marked as 1. Substituting back into (15) gives

$$\mathbb{P}[N_h \ge t] \le 8q4^{-(h+t)} + 16qt4^{-(2h+t)}. \tag{16}$$

We may assume that $t \le 2^h$, since for $t > 2^h$ we trivially have $\mathbb{P}[N_h \ge t] = 0$. Hence, the above bound simplifies to

$$\mathbb{P}[N_h \ge t] \le 8q4^{-(h+t)} + 16q4^{-(h+t)}2^{-h} \le 4^{-(h+t)}, \tag{17}$$

since $h \ge 2$ implies $2^{-h} \le \frac{1}{4}$, and we have assumed $q \le \frac{1}{12}$. ◀

▶ **Lemma 10** (Sub-Exponential Behavior for Binary Tree Paths). *Under the setup of Lemma 9, N_h satisfies $\mathbb{E}[e^{\lambda N_h}] \le 1 + 4^{-h}$ for $\lambda \le \log 2$. In addition, for any integer $h_{\max} \ge 1$, if $N_1, \dots, N_{h_{\max}}$ are independent random variables with the same distribution as N_h for the specified height, then $N = \sum_{h=1}^{h_{\max}} N_h$ satisfies $\mathbb{E}[e^{\lambda N}] \le 2$ for $\lambda \le \log 2$.*

Proof. We seek to upper bound $\mathbb{E}[e^{\lambda N_h}] = \sum_{t=0}^{\infty} \mathbb{P}[N_h = t] e^{\lambda t}$. Separating out the $t = 0$ term and applying Lemma 9, we obtain $\mathbb{E}[e^{\lambda N_h}] \le 1 + \sum_{t=1}^{\infty} 4^{-(h+t)} e^{\lambda t} = 1 + 4^{-h} \sum_{t=1}^{\infty} e^{(\lambda - \log 4)t} = 1 + 4^{-h} \cdot \frac{e^{\lambda}}{4 - e^{\lambda}}$ for $\lambda \in [0, \log 4)$. In particular, if $\lambda \le \log 2$, then $\mathbb{E}[e^{\lambda N_h}] \le 1 + 4^{-h}$.

For the second part, we again consider $\lambda \le \log 2$, and use the independence assumption to write $\mathbb{E}[e^{\lambda N}] = \prod_{h=1}^{h_{\max}} \mathbb{E}[e^{\lambda N_h}] \le \prod_{h=1}^{h_{\max}} \left(1 + 4^{-h}\right) \le e^{\sum_{h=1}^{h_{\max}} 4^{-h}} \le e^{1/3} \le 2$. ◀

We now consider the following decomposition of N_{leaf}:

$$N_{\text{leaf}} = N'_{\text{leaf}} + N''_{\text{leaf}}, \tag{18}$$

where N'_{leaf} counts the reached non-defective leaf nodes having at least one defective ancestor, and N''_{leaf} counts the reached non-defective leaf nodes having all non-defective ancestors. In the case of a single defective item (i.e., $k = 1$), we claim that N'_{leaf} has the same distribution as $2N$, with N given in Lemma 10 for a suitably-chosen value of $h_{\max}$ and $q = \frac{1}{2}$. To see this, we identify the leaf nodes in Lemma 10 with nodes at the *second-last* level of the tree illustrated in Figure 1, and observe that the factor of 2 arises since every such node produces two children at the final level (hence contributing to N'_{leaf}) when placed in a positive test.

In the case of k defective items, some care is needed, as the relevant sets of leaves associated with the k defective paths may overlap, potentially creating complicated dependencies between the associated random variables. However, if we take the definition of N in Lemma 10 and remove some leaves from the count, the quantity $\mathbb{E}[e^{\lambda N}]$ can only decrease further, so the conclusion $\mathbb{E}[e^{\lambda N}] \le 2$ remains valid. Hence, by counting any overlapping leaves only once (in an otherwise arbitrary manner), we can form k independent random variables $N^{(1)}, \dots, N^{(k)}$ satisfying $2 \sum_{i=1}^{k} N^{(i)} \overset{\text{d}}{=} N'_{\text{leaf}}$ and $\mathbb{E}[e^{\lambda N^{(i)}}] \le 2$. In addition, since there are k nodes at level $\ell_{\min} = \log_2 k$, we can similarly form $k' \le k$ independent random variables $N^{(k+1)}, \dots, N^{(k+k')}$ satisfying $\sum_{i=k+1}^{k+k'} N^{(i)} \overset{\text{d}}{=} N''_{\text{leaf}}$ and $\mathbb{E}[e^{\lambda N^{(i)}}] \le 2$.

Thus, by (18), N_{leaf} is the sum of at most $2k$ independent sub-exponential random variables. Again applying a standard concentration bound [43, Prop. 5.16], it follows that

$$\mathbb{P}[N_{\text{leaf}} \ge \mathbb{E}[N_{\text{leaf}} \,|\, \mathcal{T}_S] + t \,|\, \mathcal{T}_S] \le e^{-\Omega(\min\{t^2/k, t\})}, \tag{19}$$

from which Lemma 8 follows upon setting $t = \Theta(k)$ and using the fact that $\mathbb{E}[N_{\text{leaf}} \,|\, \mathcal{T}_S] = O(k)$ (see Lemma 5).

2.3.4 Analysis of the final level

Recall that at the final level, we perform $C' \log k$ independent sequences of $2k$ tests, with each item being randomly placed in one test in each sequence. We study the error probability conditioned on the high-probability event that $N_{\text{leaf}} = O(k)$ (see Lemma 8)

For a given non-defective item and a given sequence of $2k$ tests, the probability of colliding with any defective item is at most $\frac{1}{2}$, similarly to Lemma 2. Due to the $C' \log k$ repetitions, for any fixed $c' > 0$, there exists a choice of C' yielding $O(k^{-c'})$ probability of a given non-defective appearing only in positive tests. By a union bound over the $N_{\text{leaf}} = O(k)$ non-defectives at the final level, we find that the estimate $\widehat{S}$ equals S with (conditional) probability $1 - O(k^{1-c'})$.

2.3.5 Number of tests, error probability, decoding time, and storage

The claims of Theorem 1 are established as follows:

- *Number of tests:* As stated in Section 2.1, the number of tests is $t = Ck\log_2\frac{n}{k} + 2C'k\log k$, which behaves as $O\big(k\log\frac{n}{k} + k\log k\big) = O(k\log n)$.
- *Error probability*: The concentration bounds on N_{leaf} and N_{total} (see Lemmas 6 and 8) hold with probability $1 - e^{-\Omega(k)}$ and $1 - e^{-\Omega(k\log\frac{n}{k})}$ respectively, and we treat their complements as error events. These terms are dominated by the final stage, which incurs $O(k^{1-c'})$ failure probability; setting $c' = c+1$ gives the $O(k^{-c})$ behavior stated in Theorem 1.
- *Decoding time*: We claim that conditioned on the the high-probability events $N_{\text{total}} = O\big(k\log\frac{n}{k}\big)$ and $N_{\text{leaf}} = O(k)$, the decoding time is $O\big(k\log\frac{n}{k} + k\log k\big) = O(k\log n)$. The first term comes from considering all levels except the last, since it takes $O(1)$ time to check whether each node associated with N_{total} is in a positive or negative test. While N_{total} only counts the non-defective nodes, the number of defective nodes also trivially behaves as $O\big(k\log\frac{n}{k}\big)$. At the last level, for each of the $k + N_{\text{leaf}} = O(k)$ relevant leaf nodes, we perform $O(\log k)$ such checks for a total time of $O(k\log k)$.
- *Storage*: At the ℓ-th level, we need to store 2^ℓ integers indicating the test associated with each of the 2^ℓ nodes. Hence, excluding the last level, we need to store $\sum_{\ell=\log_2 k}^{\log_2 n-1} 2^\ell = O(n)$ integers in $\{1,\dots,2k\}$, or $O(n\log k)$ bits. Similarly, the $O(\log k)$ independent sequences of tests at the final level amount to storing $O(n\log k)$ integers, or $O(n(\log k)^2)$ bits. In addition, under the high-probability event $N_{\text{total}} = O\big(k\log\frac{n}{k}\big) = O(k\log n)$, the storage of the possibly defective set requires $O(k\log n)$ integers, or $O(k\log^2 n)$ bits. Since $k \le n$, this is no higher than $O(n\log^2 k)$.

3 Storage Reductions via Hashing

A notable weakness of Theorem 1 is that the storage required at the decoder is higher than linear in the number of items. In this section, we present a variant of our algorithm with considerably lower storage that attains similar guarantees to Theorem 1.

The idea is to interpret the mappings $\{1,\dots,2^\ell\} \to \{1,\dots,Ck\}$ at each level as hash functions. Since the high storage in Theorem 1 comes from explicitly storing the corresponding $2^\ell = O(n)$ values, the key to reducing the overall storage is to use lower-storage hash families. The reduced storage comes at the expense of reduced independence between different hash values (e.g., only r-wise independence for some $r \ll n$), and the proof of Theorem 1 utilizes full independence. The modified algorithm in this section uses $\Theta(\log k)$-wise independent hash families, and we leave open the question of whether similar recovery guarantees can be attained with only $O(1)$-wise independence.

A second modification to the algorithm is that in order to attain the behavior $O(k^{-c})$ in the error probability similarly to Theorem 1, we consider the use of $\widetilde{C} \ge 1$ independent repetitions at each level $\ell = \ell_{\min},\dots,\log_2 n - 1$, in the same way that we already used repetitions at the final level. This means that at each level, there are $\widetilde{C}$ sequences of Ck tests (each with a different and independent hash function), and a node is only considered to be possibly defective at a given level if *all* of its associated $\widetilde{C}$ tests are positive.

▶ Remark. This idea of using low-storage hash functions is reminiscent of the Bloom filter data structure [7]. A direct approach to transferring Bloom filters to group testing is to perform $L = O(\log n)$ hashes of each item into one of $t = O(k\log n)$ tests, and then estimate the defective set to be the set of items only included in positive tests [4, Sec. 1.7]. By checking the L test outcomes associated with each item, the defective set can be identified with low

storage (depending on the hash family properties) under the preceding scaling laws. However, a drawback of this direct approach is that checking all items separately takes $\Omega(n)$ time. Our algorithm circumvents this via the binary splitting approach.

3.1 Statement of Result

With the above-described modifications to the algorithm, we have the following counterpart to Theorem 1, which formalizes and generalizes Main Result 2.

▶ **Theorem 11** (Algorithmic Guarantees with Reduced Storage). *Let S be a fixed (defective) subset of $\{1,\dots,n\}$ of cardinality k. For any constant $c > 0$, there exist choices of C, $\widetilde{C}$, and C' such that the above-described group testing algorithm adapted from Section 2.1, with a $\Theta(\log k)$-wise independent hash family and $\widetilde{C}$ independent repetitions per level, yields the following with probability at least $1 - O\big(\frac{\log n}{k^c}\big)$:*

- *The returned estimate $\widehat{S}$ equals S;*
- *The algorithm runs in time $O(\mathsf{T}_{\text{hash}} k \log n)$, where T_{hash} is the evaluation time for one hash value;*
- *The algorithm uses $O(k \log n + \mathsf{S}_{\text{hash}} \log n)$ bits of storage, where S_{hash} is the number of bits of storage required for one hash function.*

In addition, the number of tests scales as $O(k \log n)$.

We briefly discuss some explicit values that can be attained for T_{hash} and S_{hash}. Supposing that n, k, and C are powers of two, we can adopt the classical approach of Wegman and Carter [44] and consider a random polynomial over the finite field $\mathrm{GF}(2^m)$, where $m \in \{\log_2 k, \dots, \log_2 n\}$ (depending on the level). In this case, one attains r-wise independence while storing r elements of $\mathrm{GF}(2^m)$ (or $O(r \log n)$ bits), and performing $O(r)$ additions and multiplications in $\mathrm{GF}(2^m)$ to evaluate the hash. As a result, with $r = \Theta(\log k)$, we get

$$\mathsf{T}_{\text{hash}} = O(\log k), \qquad \mathsf{S}_{\text{hash}} = O(\log k \cdot \log n) \tag{20}$$

under the assumption that operations in $\mathrm{GF}(2^m)$ can be performed in constant time. Hence, Theorem 11 gives $O(k \log k \cdot \log n)$ decoding time and $O(k \log n + \log k \cdot \log^2 n)$ storage. Different trade-offs can also be attained using more recent hash families that can attain r-wise independence with an evaluation time significantly less than r [40, 41].

The error probability of $O\big(\frac{\log n}{k^c}\big)$ is slightly worse than the $O(k^{-c})$ scaling of Theorem 1; in particular, $o(1)$ error probability is only guaranteed when $k = (\log n)^{\Omega(1)}$. We expect that this requirement could be avoided via a refined analysis, e.g., by characterizing the behavior of the random variables N_{total} and N_{leaf} used in the proof of Theorem 1. In addition, the following variants of Theorem 11 can already be attained with almost no additional effort:

- We can allow $c = \omega(1)$ in the statement of Theorem 11, but at the expense of the number of tests and decoding time increasing by a multiplicative $\Theta(c)$ factor.
- It is straightforward to show that $\mathbb{E}[N_{\text{total}}] = O\big(k \log \frac{n}{k}\big)$ and $\mathbb{E}[N_{\text{leaf}}] = O(k)$, and one can use Markov's inequality to deduce that $N_{\text{total}} = O\big(k^{1+c} \log \frac{n}{k}\big)$ and $N_{\text{leaf}} = O(k^{1+c})$ with probability $1 - O(k^{-c})$. For any constant $c > 0$, this approach leads to $O(k^{-c})$ error probability with $O(k \log n)$ tests, but the decoding time increases to $O(\mathsf{T}_{\text{hash}} k^{1+c} \log n)$, and the storage increases to $O(k^{1+c} \log^2 n + \mathsf{S}_{\text{hash}} \log n)$.

In the remainder of the section, we provide the proof of Theorem 11.

3.2 Analysis

3.2.1 Auxiliary variance calculation (case $\widetilde{C} = 1$)

We first study the case $\widetilde{C} = 1$ (i.e., no repetitions), as the case $\widetilde{C} > 1$ will then follow easily.

Recall that the algorithm maintains an estimate of the possibly defective (PD) set at each level. We will give conditions under which the size of this set remains at $O(k)$ throughout the course of the algorithm. For $\ell = \ell_{\min} = \log_2 k$, we trivially have at most $k \le 4k$ PD items. We will use an induction argument to show that every level has at most $4k$ PD items, with high probability.

Consider two non-defective nodes indexed by u, v at a given level ℓ having $k' \le k$ defective nodes, let $\mathcal{D}_u, \mathcal{D}_v$ denote the respective events of hashing into a test containing one or more defectives, and let D_u, D_v be the corresponding indicator random variables. The dependence of these quantities on ℓ is left implicit. We condition on all of the test placements performed at the earlier levels, accordingly writing $\mathbb{E}_\ell[\cdot]$ and $\mathrm{Var}_\ell[\cdot]$ for the conditional expectation and conditional variance. In accordance with the above induction idea, we assume that there are at most $4k$ PD nodes at level ℓ.

▶ **Lemma 12** (Mean and Variance Bounds)**.** *Under the preceding setup and definitions, if there are at most $4k$ PD nodes at level ℓ, then we have the following when $\widetilde{C} = 1$ and $C \ge 8$:*

$$\mathbb{E}_\ell\left[\sum_u D_u\right] \le \frac{k}{2} \tag{21}$$

$$\mathrm{Var}_\ell\left[\sum_u D_u\right] = c_{\mathrm{var}} k, \tag{22}$$

where the sums are over all non-defective PD nodes at the ℓ-th level, and $c_{\mathrm{var}} > 0$ is a universal constant.

The proof is given in Appendix B. Given this result, we easily deduce the following.

▶ **Lemma 13.** *For $\widetilde{C} = 1$ and $C \ge 8$, conditioned on the ℓ-th level having at most $4k$ possibly defective (PD) nodes, the same is true at the $(\ell+1)$-th level with probability $1 - O\big(\frac{1}{k}\big)$.*

Proof. Among the PD nodes at the ℓ-th level, at most k are defective, amounting to at most $2k$ children at the next level. By Lemma 12 and Chebyshev's inequality, with probability $1 - O\big(\frac{1}{k}\big)$, at most k non-defective nodes are marked as PD, thus also amount to at most $2k$ additional children at the next level, for a total of $4k$. ◀

3.2.2 Analysis of the error probability

At the first level $\ell = \log_2 k$, we trivially have $k \le 4k$ possibly defective (PD) nodes. For $\widetilde{C} = 1$, using Lemma 13 and an induction argument, the same follows for all levels simultaneously with probability at least $1 - O(k^{-1} \log n)$. For $\widetilde{C} > 1$, we note that since we have $\widetilde{C}$ repetitions at each level and only keep the nodes whose tests are *all* positive, the $1 - O(k^{-1})$ behavior becomes $1 - O(k^{-\widetilde{C}})$ due to the independence of the repetitions. Hence, the expression $1 - O(k^{-1} \log n)$ for $\widetilde{C} = 1$ generalizes to $1 - O\big(k^{-\widetilde{C}} \log n\big)$.

The analysis of the final level in Section 2.3.4 did not rely on $h(\cdot)$ being a fully independent hash function, but rather, only relied on a collision probability of $\frac{1}{2k}$ between any two given items. Since this condition still holds for any pairwise (or higher) independent hash family, we immediately deduce the same conclusion: Conditioned on the final level having $O(k)$ nodes marked as possibly defective, and by choosing C' appropriately in the algorithm description, we attain $O(k^{1-c'})$ error probability at this level for any fixed $c' > 0$.

Combining the above and setting $\widetilde{C} = c$ and $c' = 1 + c$, we attain the desired scaling $O(k^{-c} \log n)$ in the theorem statement.

3.2.3 Number of tests, decoding time, and storage

The remaining claims of Theorem 11 are established as follows:

- *Number of tests:* The number of tests is the same as in the fully independent case, possibly with a modified implied constant if $\widetilde{C} > 1$ and/or a different choice of C is used.
- *Decoding time*: The analysis of the decoding time is similar to the fully independent case (see Section 2.3.5), but each hash takes T_{hash} time to compute. Hence, the decoding time is $O\big(\mathsf{T}_{\text{hash}} k \log n\big)$.
- *Storage*: We use $O(1)$ hash functions at each level except the last, and $O(\log k)$ hash functions at the final level, for a total of $O\big(\log \frac{n}{k} + \log k\big) = O(\log n)$, requiring $O(\mathsf{S}_{\text{hash}} \log n)$ storage. In addition, under the high-probability event that there are $O(k)$ possibly defective nodes at each level, their storage requires $O(k)$ integers, or $O(k \log n)$ bits. Hence, the total storage is $O(k \log n + \mathsf{S}_{\text{hash}} \log n)$.

4 Conclusion

We have presented a novel non-adaptive group testing algorithm ensuring high-probability (for-each) recovery with $O(k \log n)$ scaling in both the number of tests and decoding time. In addition, we presented a low-storage variant with similar guarantees depending on the hash family used. An immediate open question for this variant is whether similar guarantees hold for $O(1)$-wise independent hash families. In addition, even for the fully independent version, it would be of significant interest to develop a variant that is robust to random noise in the test outcomes (see Footnote 2 on Page 4).

References

1. Matthew Aldridge. The capacity of Bernoulli nonadaptive group testing. *IEEE Transactions on Information Theory*, 63(11):7142–7148, 2017.
2. Matthew Aldridge. Individual testing is optimal for nonadaptive group testing in the linear regime. *IEEE Transactions on Information Theory*, 65(4):2058–2061, April 2019.
3. Matthew Aldridge, Leonardo Baldassini, and Oliver Johnson. Group testing algorithms: Bounds and simulations. *IEEE Transactions on Information Theory*, 60(6):3671–3687, June 2014.
4. Matthew Aldridge, Oliver Johnson, and Jonathan Scarlett. Group testing: An information theory perspective. *Foundations and Trends in Communications and Information Theory*, 15(3–4):196–392, 2019.
5. George K. Atia and Venkatesh Saligrama. Boolean compressed sensing and noisy group testing. *IEEE Transactions on Information Theory*, 58(3):1880–1901, March 2012.
6. Leonardo Baldassini, Oliver Johnson, and Matthew Aldridge. The capacity of adaptive group testing. In *IEEE International Symposium on Information Theory*, pages 2676–2680, July 2013.
7. Burton H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426, 1970.
8. Steffen Bondorf, Binbin Chen, Jonathan Scarlett, Haifeng Yu, and Yuda Zhao. Cross-sender bit-mixing coding. In *International Conference on Information Processing in Sensor Networks*, 2019.

9 Steffen Bondorf, Binbin Chen, Jonathan Scarlett, Haifeng Yu, and Yuda Zhao. Sublinear-time non-adaptive group testing with $O(k \log n)$ tests via bit-mixing coding, 2019. `arXiv:1904.10102`.

10 Sheng Cai, Mohammad Jahangoshahi, Mayank Bakshi, and Sidharth Jaggi. Efficient algorithms for noisy group testing. *IEEE Transactions on Information Theory*, 63(4):2113–2136, 2017.

11 Chun Lam Chan, Sidharth Jaggi, Venkatesh Saligrama, and Samar Agnihotri. Non-adaptive group testing: Explicit bounds and novel algorithms. *IEEE Transactions on Information Theory*, 60(5):3019–3035, May 2014.

12 Mahdi Cheraghchi. Noise-resilient group testing: Limitations and constructions. In *International Symposium on Fundamentals of Computation Theory*, pages 62–73, 2009.

13 Mahdi Cheraghchi and Vasileios Nakos. Combinatorial group testing and sparse recovery schemes with near-optimal decoding time, 2020. `arXiv:2006.08420`.

14 Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. Pattern matching with don't cares and few errors. *Journal of Computer and System Sciences*, 76(2):115–124, 2010.

15 Amin Coja-Oghlan, Oliver Gebhard, Max Hahn-Klimroth, and Philipp Loick. Information-theoretic and algorithmic thresholds for group testing. In *International Colloquium on Automata, Languages and Programming*, 2019.

16 Amin Coja-Oghlan, Oliver Gebhard, Max Hahn-Klimroth, and Philipp Loick. Optimal group testing, 2020. `arXiv:1911.02287`.

17 Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

18 Graham Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. *ACM Transactions on Database Systems*, 30(1):249–278, March 2005.

19 Robert Dorfman. The detection of defective members of large populations. *Annals of Mathematical Statistics*, 14(4):436–440, 1943.

20 Dingzhu Du and Frank K. Hwang. *Combinatorial group testing and its applications*, volume 12. World Scientific, 2000.

21 Meyer Dwass. The total progeny in a branching process and a related random walk. *Journal of Applied Probability*, 6(3):682–686, 1969.

22 Arkadii G. D'yachkov and Vladimir V. Rykov. A survey of superimposed code theory. *Problems of Control and Information*, 12(4):1–13, 1983.

23 Arkady D'yachkov, Ilya Vorobyev, Nikita Polianskii, and Vladislav Shchukin. Bounds on the rate of superimposed codes. In *IEEE International Symposium on Information Theory*, June 2014.

24 William Feller. *An introduction to probability theory and its applications*. Wiley, 1957.

25 Antonio Fernández Anta, Miguel A. Mosteiro, and Jorge Ramón Muñoz. Unbounded contention resolution in multiple-access channels. In *Distributed Computing*, volume 6950, pages 225–236. Springer Berlin Heidelberg, 2011.

26 Anna C. Gilbert, Mark A. Iwen, and Martin J. Strauss. Group testing and sparse signal recovery. In *Asilomar Conference on Signals, Systems, and Computers*, pages 1059–1063, October 2008.

27 Anna C. Gilbert, Martin J. Strauss, Joel A. Tropp, and Roman Vershynin. One sketch for all: Fast algorithms for compressed sensing. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 237–246, New York, 2007.

28 Catherine A. Hogan, Malaya K. Sahoo, and Benjamin A. Pinsky. Sample pooling as a strategy to detect community transmission of SARS-CoV-2. *Journal of the American Medical Association*, April 2020.

29 Frank K. Hwang. A method for detecting all defective members in a population by group testing. *Journal of the American Statistical Association*, 67(339):605–608, 1972.

30 Huseyin A. Inan, Peter Kairouz, Mary Wootters, and Ayfer Ozgur. On the optimality of the Kautz-Singleton construction in probabilistic group testing. *IEEE Transactions on Information Theory*, 65(9):5592–5603, September 2019.

31 Piotr Indyk, Hung Q. Ngo, and Atri Rudra. Efficiently decodable non-adaptive group testing. In *ACM-SIAM Symposium on Discrete Algorithms*, 2010.

32 Piotr Indyk and Eric Price. K-median clustering, model-based compressive sensing, and sparse recovery for earth mover distance. In *ACM Symposium on Theory of Computing*, pages 627–636, 2011.

33 Oliver Johnson, Matthew Aldridge, and Jonathan Scarlett. Performance of group testing algorithms with near-constant tests-per-item. *IEEE Transactions on Information Theory*, 65(2):707–723, February 2019.

34 Kangwook Lee, Ramtin Pedarsani, and Kannan Ramchandran. SAFFRON: A fast, efficient, and robust framework for group testing based on sparse-graph codes. *IEEE Transactions on Signal Processing*, 67(17):4649–4664, September 2019.

35 M. B. Malyutov and P. S. Mateev. Screening designs for non-symmetric response function. *Mathematical Notes of the Academy of Sciences of the USSR*, 29:109–127, 1980.

36 Mikhail B. Malyutov. The separating property of random matrices. *Mathematical Notes of the Academy of Sciences of the USSR*, 23(1):84–91, 1978.

37 Hung Q. Ngo, Ely Porat, and Atri Rudra. Efficiently decodable error-correcting list disjunct matrices and applications. In *International Colloquium on Automata, Languages and Programming*, 2011.

38 Ely Porat and Amir Rothschild. Explicit nonadaptive combinatorial group testing schemes. *IEEE Transactions on Information Theory*, 57(12):7982–7989, 2011.

39 Jonathan Scarlett and Volkan Cevher. Phase transitions in group testing. In *ACM-SIAM Symposium on Discrete Algorithms*, 2016.

40 Alan Siegel. On universal classes of fast high performance hash functions, their time-space tradeoff, and their applications. In *IEEE Symposium on Foundations of Computer Science*, 1989.

41 Mikkel Thorup. Fast and powerful hashing using tabulation. *Communications of the ACM*, 60(7):94–101, 2017.

42 Claudio M Verdun, Tim Fuchs, Pavol Harar, Dennis Elbrächter, David S Fischer, Julius Berner, Philipp Grohs, Fabian J Theis, and Felix Krahmer. Group testing for SARS-CoV-2 allows for up to 10-fold efficiency increase across realistic scenarios and testing strategies. URL: `https://www.medrxiv.org/content/10.1101/2020.04.30.20085290v2`, 2020.

43 Roman Vershynin. Introduction to the non-asymptotic analysis of random matrices, 2010. `arXiv:1011.3027`.

44 Mark N. Wegman and J. Lawrence Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22(3):265–279, 1981.

45 Idan Yelin, Noga Aharony, Einat Shaer-Tamar, Amir Argoetti, Esther Messer, Dina Berenbaum, Einat Shafran, Areen Kuzli, Nagam Gandali, Tamar Hashimshony, Yael Mandel-Gutfreund, Michael Halberthal, Yuval Geffen, Moran Szwarcwort-Cohen, and Roy Kishony. Evaluation of COVID-19 RT-qPCR test in multi-sample pools. URL: `https://www.medrxiv.org/content/early/2020/03/27/2020.03.26.20039438`, 2020.

A Discussion on the SAFFRON Algorithm

While the SAFFRON algorithm [34] is based on sparse-graph codes, we find it most instructive to compare against the simplified singleton-only version [4, Sec. 5.4], as the more sophisticated version does not attain better scaling laws (though it may attain better constant factors).

Singleton-only SAFFRON is briefly outlined as follows. One forms $O(k \log k)$ "bundles" of tests of size $2\log_2 n$ each, and assigns each item to any given bundle with probability $O\big(\frac{1}{k}\big)$. If a given item is assigned to a given bundle, then its $(\log_2 n)$-bit description is encoded into the first $\log_2 n$ tests (i.e., the item is included in the test if and only if its bit description contains a 1 at the corresponding location), and its bit-wise complement is encoded into the last $\log_2 n$ tests. The following decoding procedure ensures the identification of any defective item for which there exists a bundle in which it is included without any other defective items:

- For each bundle, check whether the first $\log_2 n$ test outcomes equal the bit-wise negation of the second $\log_2 n$ outcomes.
- If so, add the item with bit representation given by the first $\log_2 n$ outcomes into the defective set estimate.

Due to the first step, the second step will never erroneously be performed on a bundle without defective items, nor on a bundle with multiple defective items.

In [34], a decoding time of $O(k \log k \cdot \log n)$ is stated under the assumption that reading the $2 \log_2 n$ bits takes $O(\log n)$ time. However, if the associated $2 \log_2 n$ tests are stored in memory as two "words" of length $\log_2 n$ each, then a word-RAM model of computation only incurs $O(1)$ time per bundle, or $O(k \log k)$ time overall.

We also note that SAFFRON only requires $O(k \log k \cdot \log n)$ bits of storage, amounting to negligible storage overhead beyond the test outcomes themselves. This is because the only item indices stored are those added to the estimate of the defective set, and there are at most k such indices (or $k \log_2 n$ bits) due to the fact that SAFFRON makes no false positives.

B Proof of Lemma 12 (Mean and Variance Bounds)

For ease of notation, we leave the subscripts $(\cdot)_\ell$ implicit throughout the proof, but the associated conditioning is understood to apply to all probabilities, expectations, variance terms, and so on.

We first prove (21). The event $\mathcal{D}_u$ occurs if u is hashed into the same bin as any of the $k' \le k$ defective nodes. Since we are hashing into $\{1, \dots, Ck\}$ and the hash family is (at least) pairwise independent, each collision occurs with probability $\frac{1}{Ck}$. Hence, by the union bound, u is in a positive test with probability at most $\frac{1}{C}$, and (21) follows from the assumption that there are at most $4k$ PD nodes and $C \ge 8$.

As for (22), we first characterize $\mathrm{Cov}[D_u, D_v]$, writing

$$\begin{aligned} \mathrm{Cov}[D_u, D_v] &= \mathbb{E}[D_u D_v] - \mathbb{E}[D_u]\mathbb{E}[D_v] && (23)\\ &= \mathbb{P}[\mathcal{D}_u \cap \mathcal{D}_v] - \mathbb{P}[\mathcal{D}_u]\mathbb{P}[\mathcal{D}_v] && (24)\\ &= \mathbb{P}[\mathcal{D}_u] + \mathbb{P}[\mathcal{D}_v] - \mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v] - \mathbb{P}[\mathcal{D}_u]\mathbb{P}[\mathcal{D}_v]. && (25) \end{aligned}$$

We proceed by bounding $\mathbb{P}[\mathcal{D}_u]$ (the same bound holds for $\mathbb{P}[\mathcal{D}_v]$) and $\mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v]$ separately.

Probability of the individual event

Fix a non-defective node u. Let $h(\cdot)$ denote the random hash function with output values in $\{1, \dots, Ck\}$, and for each defective node indexed by $i \in \{1, \dots, k'\}$ (where k' is the total number of defective nodes at the level under consideration), let B_i be the "bad" event that $h(i) = h(u)$. We apply the inclusion-exclusion principle, which is written in terms of the following quantities for $j = 1, \dots, k'$:

$$T_j = \sum_{1 < i_1 < \dots < i_j < k'} \mathbb{P}[B_{i_1} \cap \dots \cap B_{i_j}]. \tag{26}$$

If the hash function is $(j+1)$-wise independent, this simplifies to

$$T_j = \binom{k'}{j} \left(\frac{1}{Ck}\right)^j. \tag{27}$$

Hence, if the hash function is $(j_{\max}+1)$-wise independent for some $j_{\max}$, then the inclusion-exclusion principle gives

$$\mathbb{P}[\mathcal{D}_u] = \mathbb{P}\Big[\bigcup_{i=1,\dots,k'} B_i\Big] \tag{28}$$

$$\leq \sum_{j=1}^{j_{\max}} (-1)^{j+1} T_j \tag{29}$$

$$= \sum_{j=1}^{j_{\max}} \binom{k'}{j} (-1)^{j+1} \Big(\frac{1}{Ck}\Big)^j \tag{30}$$

for odd-valued $j_{\max}$, and the reverse inequality for even-valued $j_{\max}$. Using the fact that $\sum_{j=1}^{k'} \binom{k'}{j}(-1)^{j+1}\big(\frac{1}{Ck}\big)^j = 1 - \big(1-\frac{1}{Ck}\big)^{k'}$, we can write (30) as

$$\mathbb{P}[\mathcal{D}_u] \leq 1 - \Big(1-\frac{1}{Ck}\Big)^{k'} - \sum_{j_{\max}+1}^{k'} \binom{k'}{j} (-1)^{j+1} \Big(\frac{1}{Ck}\Big)^j. \tag{31}$$

The final term can then be bounded as follows in absolute value:

$$\Big|\sum_{j_{\max}+1}^{k'} \binom{k'}{j} (-1)^{j+1} \Big(\frac{1}{Ck}\Big)^j\Big| \leq \sum_{j_{\max}+1}^{k'} \binom{k'}{j} \Big(\frac{1}{Ck}\Big)^j \tag{32}$$

$$\leq \sum_{j_{\max}+1}^{\infty} \Big(\frac{1}{C}\Big)^j \tag{33}$$

$$= \frac{(1/C)^{j_{\max}+1}}{1-1/C}, \tag{34}$$

where (33) uses $\binom{k'}{j} \leq (k')^j$ and $k' \leq k$, and (34) applies the geometric series formula. Assuming $C \geq 2$, we can further upper bound the above expression by $(1/C)^{j_{\max}}$, and hence by any target value δ_0 provided that $j_{\max} \geq \log_C \frac{1}{\delta_0}$. Recall also that (31) is reversed for even-valued $j_{\max}$, so loosening the preceding requirement to $j_{\max} \geq \lceil \log_C \frac{1}{\delta_0} \rceil + 1$ gives

$$1 - \Big(1-\frac{1}{Ck}\Big)^{k'} - \delta_0 \leq \mathbb{P}[\mathcal{D}_u] \leq 1 - \Big(1-\frac{1}{Ck}\Big)^{k'} + \delta_0. \tag{35}$$

Since u is arbitrary, the same bound also holds for $\mathbb{P}[\mathcal{D}_v]$.

Probability of the union of two events

We can decompose $\mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v]$ as follows:

$$\mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v] = \Big(1-\frac{1}{Ck}\Big)\mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v \,|\, h(u) \neq h(v)] + \frac{1}{Ck}\mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v \,|\, h(u) = h(v)] \tag{36}$$

$$= \mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v \,|\, h(u) \neq h(v)] + O\Big(\frac{1}{k}\Big). \tag{37}$$

Hence, we focus on the case $h(u) \neq h(v)$ in the following. Similar to the above, let B'_i be the "bad" event that $h(i) \in \{h(u), h(v)\}$, and define

$$T'_j = \sum_{1<i_1<\dots<i_j<k'} \mathbb{P}[B'_{i_1} \cap \dots \cap B'_{i_j} \,|\, h(u) \neq h(v)]. \tag{38}$$

If the hash function is $(j+2)$-wise independent, this simplifies to

$$T'_j = \binom{k'}{j}\Big(\frac{2}{Ck}\Big)^j, \tag{39}$$

where the factor of two comes from the possibility of colliding with either u or v. Following the same argument as above, we find that if (i) $j_{\max} \geq \lceil \log_{C/2} \frac{1}{\delta_0} \rceil + 1$, (ii) $C \geq 4$, and (iii) the hash function is $(j_{\max}+2)$-wise independent, then the following analog of (35) holds:

$$1 - \Big(1 - \frac{2}{Ck}\Big)^{k'} - \delta_0 \leq \mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v \,|\, h(u) \neq h(v)] \leq 1 - \Big(1 - \frac{2}{Ck}\Big)^{k'} + \delta_0. \tag{40}$$

Combining and simplifying

Setting $\delta_0 = \frac{1}{k}$ and combining the above findings, we deduce that for a $\Theta(\log k)$-wise independent hash,

$$\mathbb{P}[\mathcal{D}_u] = 1 - \Big(1 - \frac{1}{Ck}\Big)^{k'} + O\Big(\frac{1}{k}\Big), \tag{41}$$

$$\mathbb{P}[\mathcal{D}_u \cup \mathcal{D}_v] = 1 - \Big(1 - \frac{2}{Ck}\Big)^{k'} + O\Big(\frac{1}{k}\Big). \tag{42}$$

The idea in the following is to approximate $1 - \frac{\nu}{Ck} \approx e^{-\frac{1}{Ck}}$ for $\nu = 1, 2$, and substitute into (25). To make this more precise, we use the fact that $k' \leq k$ to write

$$\Big(1 - \frac{1}{Ck}\Big)^{k'} = \Big(e^{-\frac{1}{Ck} + O\left(\frac{1}{k^2}\right)}\Big)^{k'} \tag{43}$$

$$= e^{-\frac{k'}{Ck} + O\left(\frac{1}{k}\right)} \tag{44}$$

$$= e^{-\frac{k'}{Ck}} + O\Big(\frac{1}{k}\Big). \tag{45}$$

Applying a similar argument to $\big(1 - \frac{2}{Ck}\big)^{k'}$ and substituting into (25), we obtain

$$\mathrm{Cov}[D_u, D_v] = 2\Big(1 - e^{-\frac{k'}{Ck}}\Big) - \Big(1 - e^{-\frac{2k'}{Ck}}\Big) - \Big(1 - e^{-\frac{k'}{Ck}}\Big)^2 + O\Big(\frac{1}{k}\Big) \tag{46}$$

$$= O\Big(\frac{1}{k}\Big), \tag{47}$$

since the first three terms cancel upon expanding the square. The proof is concluded by writing

$$\mathrm{Var}\Big[\sum_u D_u\Big] = \sum_u \mathrm{Var}[D_u] + \sum_{u \neq v} \mathrm{Cov}[D_u, D_v] = O(k) \tag{48}$$

since $\mathrm{Var}[D_u] \leq \mathbb{E}[D_u] \leq \frac{1}{C}$, and there are at most $4k$ values of u by assumption.

Maximum Shallow Clique Minors in Preferential Attachment Graphs Have Polylogarithmic Size

Jan Dreier
Department of Computer Science, RWTH Aachen University, Germany
https://tcs.rwth-aachen.de/~dreier
dreier@cs.rwth-aachen.de

Philipp Kuinke
Department of Computer Science, RWTH Aachen University, Germany
https://tcs.rwth-aachen.de/~kuinke
kuinke@cs.rwth-aachen.de

Peter Rossmanith
Department of Computer Science, RWTH Aachen University, Germany
https://tcs.rwth-aachen.de
rossmani@cs.rwth-aachen.de

Abstract

Preferential attachment graphs are random graphs designed to mimic properties of real word networks. They are constructed by a random process that iteratively adds vertices and attaches them preferentially to vertices that already have high degree. We prove various structural asymptotic properties of this graph model. In particular, we show that the size of the largest r-shallow clique minor in G_m^n is at most $\log(n)^{O(r^2)}m^{O(r)}$. Furthermore, there exists a one-subdivided clique of size $\log(n)^{1/4}$. Therefore, preferential attachment graphs are asymptotically almost surely somewhere dense and algorithmic techniques developed for structurally sparse graph classes are not directly applicable. However, they are just barely somewhere dense. The removal of just slightly more than a polylogarithmic number of vertices asymptotically almost surely yields a graph with locally bounded treewidth.

2012 ACM Subject Classification Mathematics of computing → Random graphs

Keywords and phrases Random Graphs, Preferential Attachment, Sparsity, Somewhere Dense

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.14

Category RANDOM

1 Introduction

Large scale complex networks occur for example in the context of social-, biological, and technical networks. Even though these are vastly different environments, all these networks follow the same common laws. Many models [18, 23, 19, 8, 20] have been proposed that aim to mimic complex networks. The most prominent model might be the *preferential attachment graph model* (also called Barabási–Albert model) [3]. These graphs are created by a random process that iteratively adds new vertices and randomly connects them to already existing ones. Every time a new vertex is added, it is connected to the remaining graph via m random edges. Thereby, the probability that an edge from the new vertex to another vertex v is drawn is proportional to the current degree of v (see Section 2.3 for a more rigorous definition). The vertices are denoted by $v_1, v_2, v_3, \ldots$ in order of insertion. We denote the preferential attachment graph with n vertices and nm edges by G_m^n.

This process creates a certain imbalance: The degree of low-degree vertices is unlikely to increase and the degree of high-degree vertices is likely to increase even further. This so called "the rich get richer"-effect has has been widely recognized as a reasonable explanation


© Jan Dreier, Philipp Kuinke, and Peter Rossmanith;
licensed under Creative Commons License CC-BY



of the heavy tailed degree distribution of real networks. The preferential attachment model is particularly interesting from the point of mathematical analysis because of its simple formulation and interesting characteristics.

1.1 Previous Results

There exists a wide range of results on the asymptotic structure of preferential attachment graphs (for an overview see [31]). The radius of these graphs has been shown to be logarithmic [5, 12], which means they exhibit small world behavior [1, 21]. Various aspects of their degrees have been studied [13, 2, 24, 29, 28]. In particular, the degree distribution follows a power-law with exponent three [6]. However, unlike many real networks [32, 30], preferential attachment graphs have a vanishing clustering coefficient [7]. This result has been obtained by bounding the number of certain subgraphs in preferential attachment graphs. For example, the expected number of triangles or l-cycles in G_m^n has been shown to be $\log(n)^{\Theta(1)}$ for fixed l and m [7], which is lower than what is to be expected in real networks [22].

1.2 Our Results

In this work, we obtain new insights to the clustering behavior of preferential attachment graphs. We show that there is a cluster centered around a logarithmic number of tightly connected hubs that has previously not been discussed. On the other hand, removing these central hubs asymptotically yields a graph in which every r-neighborhood is a tree with at most one extra edge. In particular, at most two neighbors of a vertex are connected. This drastically bounds the local clustering coefficient. In summary, we show that the first logarithmically many vertices are the main source of structural complexity in preferential attachment graphs and removing them yields a locally extremely sparse graph.

We prove these results one by one. At first, we show that preferential attachment graphs contain one-subdivisions of cliques of logarithmic size. A one-subdivision is the graph obtained by replacing every edge with a path of length two.

▶ **Theorem 9.** *There exists a constant c such that for $m, n \in \mathbf{N}$, $m \geq 2$, $n \geq c$, the preferential attachment graph G_m^n contains a one-subdivided clique of size at least $\log(n)^{1/4}$ with a probability of at least $1 - e^{-\log(n)^{1/4}/c}$.*

Graph classes which contain subdivided cliques of logarithmic size cannot for example be planar or have bounded treewidth. In fact, they are *somewhere dense*, as we will discuss later on. The previous theorem is complemented by showing that removing the first polylogarithmically many vertices of a preferential attachment graph leaves a graph with a locally extremely sparse, tree-like structure.

▶ **Corollary 13.** *There exists a constant c such that for $l, n, m \in \mathbf{N}^+$ and $b = \lceil \log(n)^{cl^2} m^{cl} \rceil$ the graph $G_m^n[v_{b+1}, \ldots, v_n]$ contains a.a.s. no subgraph with up to l vertices and more edges than vertices.*

Forbidding subgraphs with more edges than vertices is quite restrictive. The previous statement states that for a fixed m, removing $\log(n)^{O(l^2)}$ vertices leads asymptotically to a graph in which all connected subgraphs with l vertices have either l or $l-1$ edges, i.e., they are either a tree or a tree with one extra edge. This means for example that the removal of $\log(n)^{O(1)}$ vertices yields asymptotically a graph with no cliques of size four. It also yields the following statement about tree-like neighborhoods in preferential attachment graphs.

▶ **Theorem 16.** *There exists a constant c such that for $r, n, m \in \mathbf{N}^+$ and $b = \lceil \log(n)^{cr^2} m^{rc} \rceil$, a.a.s. every r-neighborhood of $G_m^n[v_{b+1}, \dots, v_n]$ is either a tree or a tree with one additional edge.*

1.3 Sparsity

An often observed property of real networks is a relatively low average degree. For example in a social network, the vast majority of members only has relatively few neighbors, especially compared to the set of all potential neighbors. This leads us to a concept called *structural sparsity* [25]. Sparsity has been deeply studied and has lead to many interesting results. In particular, there exists a large body of work focusing on algorithmic applications of sparsity (for example [4, 9, 14, 16, 17]). Many graph problems that are hard for general graphs become easier on structurally sparse graph classes.

Sparsity is not a property of single graphs, but a property of graph classes. An important cornerstone of sparsity theory are *nowhere dense* and *somewhere dense* graph classes introduced by Nešetřil and Ossona de Mendez [25]. These graph classes are defined using *shallow topological clique minors*. A graph G contains an r-shallow topological clique minor of size k if a subgraph of G is isomorphic to a graph obtained by subdividing a k-clique up to $2r$ times. The size of the largest r-shallow topological clique minor of a graph G is denoted by $\omega(G \mathbin{\widetilde{\nabla}} r)$. A graph class is nowhere dense if for every radius r and every graph G in the graph class, $\omega(G \mathbin{\widetilde{\nabla}} r) \leq f(r)$ for some function $f(r)$ independent of the graph size. Conversely, if $\omega(G \mathbin{\widetilde{\nabla}} r)$ is unbounded for some r, the graph class is said to be somewhere dense. Every graph class is either nowhere or somewhere dense. For a rigorous definition of these two concepts see Section 2.4.

Nowhere dense graph classes generalize many different sparse graph properties such as bounded degree, planarity, bounded treewidth, bounded genus or bounded expansion [25]. An important algorithmic result is that every problem definable in first-order logic can be solved in almost linear time on nowhere dense graph classes [16]. On the other hand, somewhere density defines for many problems a natural barrier in tractability. The model checking problem for first-order logic on monotone somewhere dense graph classes is AW[$*$]-complete and therefore as hard as on general graphs [16].

In order to lift deep algorithmic results for sparse graph classes to random graphs, people started analyzing which random graph models are somewhere or nowhere dense *in the limit*. Roughly speaking, a random graph model is asymptotically almost surely (a.a.s.) nowhere dense if for large graphs $\omega(G \mathbin{\widetilde{\nabla}} r)$ is with probability one bounded by some function $f(r)$ independent of the graph size and a.a.s. somewhere dense if it is with probability one unbounded. If a random graph model is a.a.s. nowhere dense it is not a.a.s. somewhere dense, and vice versa. However, it is possible that a random graph model is neither a.a.s. somewhere nor a.a.s. nowhere dense. Again, for more details see Section 2.4.

Erdős–Rényi graphs, random intersection graphs, Chung–Lu graphs, and the configuration model have been classified with respect to a.a.s. somewhere and nowhere density [26, 10, 15]. This has lead directly to efficient algorithms on random graphs. Preferential attachment graphs are known not to be a.a.s. nowhere dense [10] but a complete classification remained open. In this work, we show that shallow topological clique minors have polylogarithmic size and therefore that the preferential attachment model is a.a.s. somewhere dense.

▶ **Theorem 1.** *Let $r \in \mathbf{N}^+$, $m \geq 2$. Then a.a.s. $\log(n)^{1/4} \leq \omega(G_m^n \mathbin{\widetilde{\nabla}} r) \leq \log(n)^{O(r^2)} m^{O(r)}$. In particular G_m^n is a.a.s. somewhere dense.*

This is a direct consequence of Theorem 9 and Theorem 15. Our result implies that algorithmic techniques developed for nowhere dense graph classes do not directly transfer to preferential attachment graphs. However, shallow topological clique minors have only polylogarithmic size. Hence, preferential attachment graphs are "just barely" somewhere dense. In particular, the removal of few vertices places the random graph model a.a.s. in a very sparse graph class, where all connected subgraphs of bounded diameter are "tree-like." Note that locally bounded treewidth implies nowhere denseness [25].

▶ **Theorem 17.** *Let $m \in \mathbf{N}^+$ and $g(n)\colon \mathbf{N} \to \mathbf{N}$ be a function with $g(n) = \omega(1)$. If we remove the first $\log(n)^{g(n)}$ vertices from G_m^n the remaining graph has a.a.s. locally bounded treewidth.*

2 Preliminaries

2.1 Graph Notation

We use common graph theory notation [11]. Furthermore, an *r-neighborhood* is an induced subgraph with radius at most r.

2.2 Probabilities and Random Graph Models

In this work, a random graph model (such as the preferential attachment model) is a sequence $\mathcal{G} = (\mathcal{G}_n)_{n\in\mathbf{N}}$, where $\mathcal{G}_n$ is a probability distribution over undirected simple graphs with n vertices (this means graphs have no self-loops or multi-edges). In slight abuse of notation, we also write $\mathcal{G}_n$ for the random variable which is distributed according to $\mathcal{G}_n$. This way, we can lift graph notation to notation for random variables of graphs: For example edge set and clique number of a random graph $\mathcal{G}_n$ are represented by random variables $E(\mathcal{G}_n)$ and $\omega(\mathcal{G}_n)$.

2.3 The Preferential Attachment Graph Model

In this work we focus on the *preferential attachment random graph model* which we describe in this subsection. It has been ambiguously defined in the original article by Barabási and Albert [3]. The model generates random graphs by iteratively inserting new vertices and edges. It depends on a parameter, usually denoted by m, which indicates the number of edges attached to each newly created vertex. We follow the rigorous definition of Bollobás et al. [6]: For a fixed parameter m the random process is defined by starting with a single vertex and iteratively adding vertices, thereby constructing a sequence of graphs $\bar{G}_m^1, \bar{G}_m^2, \ldots, \bar{G}_m^t$, where $\bar{G}_m^t$ has t vertices and mt edges (of which some may be self-loops). We define $d_m^t(v)$ to be the degree of vertex v in the graph $\bar{G}_m^t$. The random process for $m = 1$ works as follows. A random graph is started with one vertex v_1 that has exactly one self-loop. This graph is $\bar{G}_1^1$. We then define the graph process inductively: Given $\bar{G}_1^{t-1}$ with vertex set $\{v_1, \ldots, v_{t-1}\}$, we create $\bar{G}_1^t$ by adding a new vertex v_t together with a single edge from v_t to v_i, where i is chosen at random from $\{1, \ldots, t\}$ with

$$\Pr[i = s] = \begin{cases} \dfrac{d_1^{t-1}(v_s)}{2t-1} & 1 \le s \le t \\ \dfrac{1}{2t-1} & s = t. \end{cases}$$

This means we add an edge to a random vertex with a probability proportional to its degree at the time.

For $m > 1$, the process can be defined by merging sets of m consecutive vertices in $\bar{G}_1^{mt}$ to single vertices in $\bar{G}_m^t$ [6]. Let $v'_1, \dots, v'_{mt}$ be the vertices of $\bar{G}_1^{mt}$. The graph $\bar{G}_m^t$ with vertices $v_1, \dots, v_t$ is constructing by merging $v'_{(i-1)m+1}, \dots, v'_{im}$ into a single vertex v_i. The graph $\bar{G}_m^t$ is a multigraph. The number of edges between vertices v_i and v_j in $\bar{G}_m^t$ equals the number of edges between the corresponding sets of vertices in $\bar{G}_1^{mt}$. Self-loops are allowed. We obtain a simple random graph G_m^n from $\bar{G}_m^n$ by removing all self-loops and replacing multiple edges with a single edge.

2.4 Sparsity

At first, we define nowhere and somewhere density as a property of *graph classes* and then lift the notation to *random graph models*. There are various equivalent definitions and we use the most common definition based on shallow topological minors.

▶ **Definition 2** (Shallow topological minor [27]). *A graph H is an r-*shallow topological minor *of a graph G if a graph obtained from H by subdividing every edge up to $2r$ times is isomorphic to a subgraph of G. The set of all r-shallow topological minors of a graph G is denoted by $G \mathbin{\widetilde{\nabla}} r$. We define the maximum clique size over all shallow topological minors of G as*

$$\omega(G \mathbin{\widetilde{\nabla}} r) = \max_{H \in G \mathbin{\widetilde{\nabla}} r} \omega(H).$$

▶ **Definition 3** (Nowhere dense, somewhere dense [25]). *A graph class $\mathcal{G}$ is* nowhere dense *if there exists a function f, such that for all r and all $G \in \mathcal{G}$, $\omega(G \mathbin{\widetilde{\nabla}} r) \leq f(r)$. A graph class $\mathcal{G}$ is* somewhere dense *if for all functions f there exists an r and a $G \in \mathcal{G}$, such that $\omega(G \mathbin{\widetilde{\nabla}} r) > f(r)$.*

Observe that a graph class is somewhere dense if and only if it is not nowhere dense. We lift these notions to random graph models using the following two definitions.

▶ **Definition 4** (a.a.s. nowhere dense, a.a.s. somewhere dense). *A random graph model $\mathcal{G}$ is* a.a.s. nowhere dense *if there exists a function f such that for all r*

$$\lim_{n \to \infty} \Pr[\omega(\mathcal{G}_n \mathbin{\widetilde{\nabla}} r) \leq f(r)] = 1.$$

A random graph model $\mathcal{G}$ is a.a.s. somewhere dense *if for all functions f there is an r such that*

$$\lim_{n \to \infty} \Pr[\omega(\mathcal{G}_n \mathbin{\widetilde{\nabla}} r) > f(r)] = 1.$$

While for graph classes the concepts are complementary, a random graph model can be *neither* a.a.s. somewhere dense *nor* a.a.s. nowhere dense (e.g., if the random graph model is either the empty or the complete graph, both with a probability of $1/2$).

3 Lower Bounds

In this section we show that preferential attachment graphs are a.a.s. somewhere dense. We do so by analyzing the probability that a preferential attachment graph of size n contains a one-subdivided clique of size at least $k = \log(n)^{1/4}$ as a subgraph. Let this probability be p_n. We show that $\lim_{n \to \infty} p_n = 1$. The proof works as follows: We start with a small preferential attachment graph and pick a set of k vertices with high degree. These will be the principal vertices of the one-subdivided clique. We then add vertices to the graph according to the

preferential attachment process. A one-subdivided clique of size k arises if for every pair of principal vertices v and w, a new vertex u is added that is adjacent to both v and w. We show that after $n = k^4 2^{k^3}$ vertices have been inserted, with high probability there is at least one connecting vertex for every pair of principal vertices.

We now describe an urn experiment that illustrates for a pair of principal vertices v,w the probability that a new vertex u is connected to both u and v. This experiment has no connection to Pólya Urns and is solely used for illustrative purposes. The experiment consists of multiple rounds. In the ith round (we assume $i \geq 10$), we define an urn containing i balls, where $\sqrt{\lceil i \rceil}$ balls are red, $\sqrt{\lceil i \rceil}$ balls are blue, and the rest is black. In each round we draw two balls uniformly at random from the urn. The experiment succeeds if we draw a red and a blue ball in the same round. It is easy to see that the probability of success in the ith round equals $2(\sqrt{\lceil i \rceil}/i)^2$. We observe that eventually the experiment succeeds because

$$1 - \prod_{i=10}^{\infty} \left(1 - 2\Big(\frac{\sqrt{i}}{i}\Big)^2\right) = 1.$$

This experiment behaves similarly to the process of connecting two principal vertices. Two principal vertices are connected in round i if the vertex v_i is connected to both principal vertices. The expected degree of the first vertex in a preferential attachment graph of size i is proportional to $\sqrt{i}$. If we (naively) assume that the degrees of u and v at time i are at all times exactly $\sqrt{i}$ then a new vertex throws an edge to v or w with probability proportional to $\sqrt{i}/i$ and is connected to both with probability roughly $2(\sqrt{i}/i)^2$. Therefore, the probability that in the ith step a new vertex u connects both v and w is proportional to the probability that in the ith round of the urn experiment a red and a blue ball is drawn. Using similar arguments we show in this section that the success probability of building a one-subdivided clique also is high.

If we however alter the urn experiment and assume that in the ith round there are only about $\sqrt{i}/\log(i)$ red or blue balls we cannot guarantee success because

$$1 - \prod_{i=10}^{\infty} \left(1 - 2\Big(\frac{\sqrt{i}/\log(i)}{i}\Big)^2\right) \neq 1.$$

This means if the expected value of the degrees were just a logarithmic factor smaller then our proof would not work. This suggests that preferential attachment graphs are "just barely" a.a.s. somewhere dense. This also means we need lower bounds which guarantee that the degrees of our principal vertices are not much smaller than their expected value, e.g., at most a constant factor off. Bounds which guarantee a factor of $1/\log(i)$ would not be sufficient.

Unfortunately the probability distribution of the degree of a vertex is only well centered around its expected value if its initial degree is already large (Proposition 5). We therefore use Lemma 6 to find in a graph of size k^4 with high probability k vertices with a degree of roughly k. These vertices will be our principal vertices. For large k, their degree is centered closely around its expected value for our technique to work.

In our proof of Theorem 9 we argue that with high probability for every pair of principal vertices there will eventually be a vertex which is connected to both. One may try to prove this by showing that with high probability the degrees of the principal vertices are well centered at all times of the random process (roughly $\sqrt{i}$ in the ith round for every i) and then showing that the probability that the principal vertices will be connected under this condition is high. However, it is highly non-trivial to bound the probability that an edge is inserted during the preferential attachment process under the condition that something else will happen *afterwards*. Therefore this approach does not work easily. It is much easier to

condition under events that happen *before* in the random process. We therefore analyze the random process over time and *interleave* the concentration bounds of the principal vertices with bounds on the event that a new vertex connects a pair of principal vertices. This interleaved analysis works as follows: Let B_i be the event that the degree of the principal vertices is at least half their expected degree at time step i of the random process. We prove B_i to be likely. For j slightly larger than i, one can easily obtain a good bound on the probability that a new edge of the jth vertex is connected to a principal vertex under the assumption B_i. Our calculations work in a way where whenever we assume a new event B_i the small probability $\Pr[\bar{B}_i]$ is added to our failure probability (see Lemma 8). So if we were to assume all events $B_i, B_{i+1}, B_{i+2}, \dots$ our bound quickly becomes meaningless as the sum $\Pr[\bar{B}_i] + \Pr[\bar{B}_{i+1}] + \Pr[\bar{B}_{i+2}] + \cdots$ quickly becomes larger than one. But if we assume exponentially spaced events $B_i, B_{2i}, B_{4i}, B_{8i}, \dots$ our bound on the failure probability stays small enough and new vertices are still likely to be connected to our principal vertices, allowing us to show in Theorem 9 that G_2^n contains a large one-subdivided clique.

The results of this section build upon the following concentration bound for vertex degrees. We define $d_m^n(S) = \sum_{v \in S} d_m^n(v)$ as the summed degree of S in G_m^n.

▶ **Proposition 5** ([13], Theorem 19). *For $t, m, d \in \mathbf{N}^+$, $0 < \varepsilon \leq 1/2$, $S \subseteq \{v_1, \dots, v_t\}$ with $\Pr[d_m^t(S) = d] \neq 0$ and $d \geq \log(\log(3tm))\varepsilon^{-200}$*

$$\Pr\left[(1-\varepsilon)\sqrt{\frac{n}{t}}d < d_m^n(S) < (1+\varepsilon)\sqrt{\frac{n}{t}}d \text{ for all } n \geq t \;\middle|\; d_m^t(S) = d\right] \geq 1 - e^{-\varepsilon^{200}d}.$$

This bound is strong if the initial degree $d_m^t(S)$ is large. At first, we show that there are some vertices which have a reasonably high degree after a short number of steps.

▶ **Lemma 6.** *There exists a constant c such that for $k, m \in \mathbf{N}$, $k \geq c$ with probability at least $1 - ke^{-k/c}$ there exists a set of vertices $X \subseteq \{v_1, \dots, v_{k^2}\}$, $|X| = k$ such that $d_m^{k^4}(x) \geq mk/2$ for all $x \in X$.*

Proof. We partition the first k^2 vertices into k sets of k vertices. Let S be one of these sets. Since $|S| = k$ and every vertex has at least degree m, we know that $d_m^{k^2}(S) \geq mk$. Therefore

$$\Pr\left[d_m^{k^4}(S) \leq \frac{1}{2}mk^2\right] \leq \sum_{d=mk}^{\infty} \Pr\left[d_m^{k^2}(S) = d\right] \cdot \Pr\left[d_m^{k^4}(S) \leq \frac{1}{2}\sqrt{\frac{k^4}{k^2}}d_m^{k^2}(S) \;\middle|\; d_m^{k^2}(S) = d\right].$$

Since $k \geq c$, we can choose c sufficiently large to always guarantee $c \geq 2^{200}$ and $mk \geq 2^{200}\log\log(mk^2)$. Using $t = k^2$, $n = k^4$, and $\varepsilon = 1/2$, Proposition 5 bounds the term above by at most

$$\sum_{d=mk}^{\infty} \Pr\left[d_m^{k^2}(S) = d\right] \cdot e^{-\varepsilon^{200}d} \leq e^{-k/c}.$$

With a probability of at least $1 - ke^{-k/c}$ each of the k sets have at time k^4 a total degree of at least $mk^2/2$ by the union bound. We define x_i to be the vertex in the ith set that has the highest degree after k^4 steps. Since the set contains k vertices, x_i has a degree of at least $mk/2$. We set $X = \{x_1, \dots, x_k\}$. ◀

We now bound the probability that two principal vertices v_a, v_b become connected under the condition that they have high degree.

▶ **Lemma 7.** *We consider the preferential attachment process with $m \geq 2$. Let $k \in \mathbf{N}$ and $a, b \leq k^4$. Let B_i be the event that $d_m^i(v_a), d_m^i(v_b) \geq m\sqrt{i}/4k$. Let $A_{j,i}$ with $k^4 < j \leq i$ be the event that the first two edges of at least one of the vertices $v_j, \ldots, v_i$ are incident to v_a and v_b, respectively. Then $\Pr[\bar{A}_{i+1,2i} \mid \bar{A}_{j,i}, B_i] \leq e^{-\frac{1}{256k^2}}$ for $k^4 \leq i$ and $j \leq i$.*

Proof. Let $u > 0$. $\Pr[A_{i+u,i+u} \mid B_i]$ is the probability that vertex v_{i+u} is adjacent to both v_a and v_b under the condition that v_a and v_b have degree at least $m\sqrt{i}/4k$ at some earlier time i. When vertex v_{i+u} is inserted, the random process draws $m \geq 2$ edges from v_{i+u} to earlier vertices. The probability that some vertex is chosen equals its degree divided by the total number of edges in the graph at this time. The degree of v_a and v_b is at least $m\sqrt{i}/4k$ at this point in time. Also there is a total of at most $2(i+u)m$ edges in the graph. We can therefore bound

$$\Pr[A_{i+u,i+u} \mid B_i] \geq \Big(\frac{m\sqrt{i}/4k}{2(i+u)m}\Big)^2 = \Big(\frac{\sqrt{i}}{8(i+u)k}\Big)^2.$$

The same argument holds if we additionally assume some of the earlier vertices not to be adjacent to both v_a and v_b. Let $j < i$. Then

$$\Pr[A_{i+u,i+u} \mid \bar{A}_{j,i+u-1}, B_i] \geq \Big(\frac{\sqrt{i}}{8(i+u)k}\Big)^2.$$

We now consider the probability that no vertex in a sequence of vertices is adjacent to both v_a and v_b. The chain rule yields

$$\begin{aligned}\Pr[\bar{A}_{i+1,2i} \mid \bar{A}_{j,i}, B_i] &= \prod_{u=1}^{i} \Pr[\bar{A}_{i+u,i+u} \mid \bar{A}_{j,i+u-1}, B_i] \\ &\leq \prod_{u=1}^{i} \Big(1 - \Big(\frac{\sqrt{i}}{8(i+u)k}\Big)^2\Big) \\ &\leq \Big(1 - \Big(\frac{\sqrt{i}}{16ik}\Big)^2\Big)^i \leq \Big(1 - \frac{1}{256ik^2}\Big)^i \leq e^{-\frac{1}{256k^2}}.\end{aligned}$$

◀

Imagine a sequence of events $\mathcal{A}_0, \ldots, \mathcal{A}_{l-1}$ such that a preferential attachment graph contains a large subdivided clique if any one of these events occurs. This means it is sufficient to show that the probability $\Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{l-1}]$ is small. Assume we can only bound the probability of event $\bar{\mathcal{A}}_i$ under the condition $\mathcal{B}_i$. The following technical observation gives a good approximation of $\Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{l-1}]$ if the events $\mathcal{B}_i$ have a high probability.

▶ **Lemma 8.** *Let $\mathcal{A}_0, \ldots, \mathcal{A}_l, \mathcal{B}_0, \ldots, \mathcal{B}_l$ be events. Then*

$$\Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_l] \leq \sum_{0=1}^{l} \Pr[\bar{\mathcal{B}}_i] + \prod_{0=1}^{l} \Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1} \cap \mathcal{B}_i].$$

Proof. Let $i \leq l$. We apply the chain rule and the law of total probability. Then

$$\begin{aligned}\Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_i] &= \Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1}] \Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1}] \\ &= \Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1}] \Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1} \cap \mathcal{B}_i] \Pr[\mathcal{B}_i] \\ &\quad + \Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1}] \Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1} \cap \bar{\mathcal{B}}_i] \Pr[\bar{\mathcal{B}}_i] \\ &\leq \Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1}] \Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1} \cap \mathcal{B}_i] + \Pr[\bar{\mathcal{B}}_i].\end{aligned}$$

We can now recursively apply this inequality, and use an upper bound of 1 for all factors in front of $\Pr[\mathcal{B}_1], \ldots, \Pr[\mathcal{B}_l]$ when expanding the product, to get our claim. ◀

We now use Lemma 6 and Lemma 7 to prove the main result of this section.

▶ **Theorem 9.** *There exists a constant c such that for $m, n \in \mathbf{N}$, $m \geq 2$, $n \geq c$, the preferential attachment graph G_m^n contains a one-subdivided clique of size at least $\log(n)^{1/4}$ with a probability of at least $1 - e^{-\log(n)^{1/4}/c}$.*

Proof. Let $k \in \mathbf{N}$. We choose $n = k^4 2^{k^3}$ which implies $k \geq \log(n)^{1/4}$ for $k \geq 2$. We will prove this theorem by showing that k vertices in G_m^n are with high probability pairwise connected by a path of length two and thereby span a one-subdivided clique.

We know that if a vertex has high degree early on then their degree will be centered closely around its expected values in the future (Proposition 5). Let us therefore assume there is a set of vertices $X \subseteq \{v_1, \ldots, v_{k^2}\}$, $|X| = k$ such that $d_m^{k^4}(x) \geq mk/2$ for all $x \in X$. We call these vertices principal vertices. Since $n \geq c$, we have $k \geq \log(c)^{1/4}$. We can choose the constant c high enough such that according to Lemma 6 these principal vertices exist with a probability of at least $1 - ke^{-k/c}$.

Let us fix a pair of principal vertices v_a, v_b and show that with high probability there is a vertex that is adjacent to both v_a and v_b. The higher the degree of v_a and v_b the higher the probability that a new vertex is adjacent. As in Lemma 7, we define for $i \geq k^4$, B_i to be the event that $d_m^i(v_a), d_m^i(v_b) \geq m\sqrt{i}/4k$. Again using $k \geq \log(c)^{1/4}$, we can assume c large enough to guarantee $mk/2 \geq \log(\log(3k^4 m))2^{200}$. Since we have $d_m^{k^4}(v_a) \geq mk/2$, Proposition 5 states with $t = k^4$, $\varepsilon = 1/2$ and $d \geq mk/2$

$$\Pr\left[d_m^i(v_a) < \frac{m\sqrt{i}}{4k} = \frac{1}{2}\sqrt{\frac{i}{k^4}}mk/2\right] \leq e^{-\varepsilon^{200}mk/2} = e^{-|\Omega(k)|}.$$

This yields $\Pr[\bar{B}_i] = e^{-|\Omega(k)|}$. We further define $A_{j,i}$ with $k^4 < j \leq i$ to be the event that the first two edges of at least one of the vertices $v_j, \ldots, v_i$ is adjacent to both v_a and v_b. We will show that v_a and v_b have a connecting vertex in G_m^n by showing that $\Pr[\bar{A}_{k^4+1,n}]$ converges to zero. We divide our vertices from $k^4 + 1$ to $n = k^4 2^{k^3}$ into k^3 windows which double in size. For $0 \leq i < k^3$ we set $\bar{\mathcal{A}}_i = \bar{A}_{k^4 2^i + 1, k^4 2^{i+1}}$ and $\mathcal{B}_i = B_{k^4 2^i}$. Lemma 7 states $\Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1} \cap \mathcal{B}_i] \leq e^{-\frac{1}{256k^2}}$. By Lemma 8

$$\begin{aligned}
\Pr[\bar{A}_{k^4+1,n}] = \Pr[\bar{A}_{k^4+1,k^4 2^{k^3}}] &= \Pr[\bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{k^3-1}] \\
&\leq \sum_{i=0}^{k^3-1} \Pr[\bar{\mathcal{B}}_i] + \prod_{i=0}^{k^3-1} \Pr[\bar{\mathcal{A}}_i \mid \bar{\mathcal{A}}_0 \cap \cdots \cap \bar{\mathcal{A}}_{i-1} \cap \mathcal{B}_i] \\
&\leq \sum_{i=0}^{k^3-1} e^{-|\Omega(k)|} + \prod_{i=0}^{k^3-1} e^{-\frac{1}{256k^2}} \\
&\leq k^3 e^{-|\Omega(k)|} + e^{-\frac{k^3}{256k^2}} = k^3 e^{-|\Omega(k)|}.
\end{aligned}$$

This means that in G_m^n the probability that there exists a vertex which connects the principal vertices v_a and v_b is at least $1 - k^3 e^{-|\Omega(k)|}$. According to the union bound, the probability that for all $\binom{k}{2}$ pairs of principal vertices there exists a vertex which connects them is bounded by $1 - \binom{k}{2} k^3 e^{-|\Omega(k)|}$. In Lemma 7, only the first two edges of the connecting vertex are considered. Therefore each connecting vertex may only connect a single pair of principal vertices. This means every pair of principal vertices has a unique connecting vertex, i.e., the principal vertices span a one-subdivided clique with probability $1 - \binom{k}{2} k^3 e^{-|\Omega(k)|}$.

So far, all our calculations were based on the assumption that there are k principal vertices with reasonably high degree in the beginning. According to Lemma 6, the probability that k such vertices do not exist is at most $ke^{-k/c}$. So by law of total probability, we can add this error probability to the conditional bound to get an unconditional bound. This means that G_m^n contains no one-subdivided k clique with a probability of at most

$$\binom{k}{2}k^3e^{-|\Omega(k)|} + ke^{-k/c} = e^{-|\Omega(\log(n)^{1/4})|}. \qquad \blacktriangleleft$$

We can restate Theorem 9 as a lower bound on 1/2-shallow clique minors.

▶ **Corollary 10.** *Let $m \geq 2$. Then a.a.s. $\omega(G_m^n \mathbin{\widetilde{\nabla}} \frac{1}{2}) \geq \log(n)^{1/4}$. In particular, G_m^n is a.a.s. somewhere dense.*

4 Upper Bounds

In this section we prove polylogarithmic upper bounds on the density of shallow topological minors in preferential attachment graphs. In fact, we even show that it is sufficient to remove a polylogarithmic number of vertices to make these graphs look locally almost like trees. Our results are based on the following proposition that bounds the probability of edges occurring between fixed vertices.

▶ **Proposition 11** ([13], Lemma 10). *Let $n, m \in \mathbf{N}^+$, $n \geq 2$ and $E \subseteq \binom{v_1,\dots,v_n}{2}$. Then*

$$\Pr[E \subseteq E(G_m^n)] \leq \log(n)^{O(|E|)^2} m^{2|E|} \prod_{v_x v_y \in E} 1/\sqrt{xy}.$$

The main work is spend in Lemma 12, where we bound for $l, b \in \mathbf{N}$ the expected number of dense subgraphs of size l after removing the first b vertices of the graph. This is done by summing over all potential vertex sets $\binom{v_{b+1},\dots,v_n}{l}$ of a dense subgraph and using the previous proposition to bound the probability that these vertices induce a graph with many edges. This bound is quite strong and flexible, but its dependence on five parameters b, l, k, n, m makes it a bit unwieldy. We simplify it to obtain statements abound the size of r-shallow minors or the density of local regions.

▶ **Lemma 12.** *Let $b, l, k, n, m \in \mathbf{N}^+$ with $n \geq 2$. The expected number of subgraphs of $G_m^n[v_{b+1}, \dots, v_n]$ with at most l vertices and k more edges than vertices is at most $\log(n)^{O(l+k)^2} m^{O(l+k)} b^{-k}$.*

Proof. Let H be a graph with at most l vertices and k more edges than vertices. Let p be the expected number of subgraphs of $G_m^n[v_{b+1}, \dots, v_n]$ that are isomorphic to H. We want to find an upper bound for p. We iteratively remove every degree-one vertex from H. The resulting graph H' still has at most l vertices and k more edges than vertices. Let p' the expected number of subgraphs of $G_m^n[v_{b+1}, \dots, v_n]$ isomorphic to H'. Since H' is a subgraph of H, we have $p \leq p'$.

Let $V(H') = \{u_1, \dots, u_\gamma\}$ with $\gamma \leq l$ and let $\delta_1, \dots, \delta_\gamma$ be the degree sequence of $V(H')$. We state two inequalities that we will use later. Inequality (1) follows from $\delta_i \geq 2$ and (2) holds since the number of edges in H' is $\gamma + k = \sum_{i=1}^{\gamma} \delta_i/2$.

$$\sum_{y_i=b+1}^{n} \frac{1}{\sqrt{y_i}^{\delta_i}} \leq \int_b^n \frac{1}{\sqrt{x}^{\delta_i}} dx \leq 2b^{1-\delta_i/2} \log n \tag{1}$$

$$\sum_{i=1}^{\gamma} (-\delta_i/2 + 1) = -k \tag{2}$$

For integers $b < x_1, \dots, x_\gamma \leq n$, we consider an embedding of H' into the graph $G_m^n[v_{b+1}, \dots, v_n]$ that maps u_i to v_{x_i} (for $1 \leq i \leq \gamma$). According to Proposition 11, the probability that this embedding of H' is a subgraph of G_m^n is at most

$$\log(n)^{O(l+k)^2} m^{O(l+k)} \prod_{i=1}^{\gamma} \frac{1}{\sqrt{x_i}^{\delta_i}}.$$

We use the union bound over all valid embeddings, (1), and (2) to bound

$$\begin{aligned}
p \leq p' &\leq \sum_{x_1=b+1}^{n} \cdots \sum_{x_\gamma=b+1}^{n} \log(n)^{O(l+k)^2} m^{O(l+k)} \prod_{i=1}^{\gamma} \frac{1}{\sqrt{x_i}^{\delta_i}} \\
&= \log(n)^{O(l+k)^2} m^{O(l+k)} \sum_{x_1=b+1}^{n} \frac{1}{\sqrt{x_1}^{\delta_1}} \cdots \sum_{x_\gamma=b+1}^{n} \frac{1}{\sqrt{x_\gamma}^{\delta_\gamma}} \\
&\overset{(1)}{\leq} \log(n)^{O(l+k)^2} m^{O(l+k)} \prod_{i=1}^{\gamma} 2b^{1-\delta_i/2} \log(n) \\
&\overset{(2)}{\leq} \log(n)^{O(l+k)^2} m^{O(l+k)} 2^\gamma \log(n)^\gamma b^{-k} \leq \log(n)^{O(l+k)^2} m^{O(l+k)} b^{-k}.
\end{aligned}$$

For an arbitrary but fixed graph H with at most l vertices and k more edges than vertices we have bounded the expected number of valid embeddings p. There are at most 2^{l^2} graphs with l vertices. Therefore, the expected number of subgraphs of $G_m^n[v_{b+1}, \dots, v_n]$ with at most l vertices and k more edges than vertices is at most $2^{l^2} \log(n)^{O(l+k)^2} m^{O(l+k)} b^{-k} = \log(n)^{O(l+k)^2} m^{O(l+k)} b^{-k}$. ◀

The following corollary simplifies the last lemma by choosing the specified value for b.

▶ **Corollary 13.** *There exists a constant c such that for $l, n, m \in \mathbf{N}^+$ and $b = \lceil \log(n)^{cl^2} m^{cl} \rceil$ the graph $G_m^n[v_{b+1}, \dots, v_n]$ contains a.a.s. no subgraph with up to l vertices and more edges than vertices.*

▶ **Lemma 14.** *There is a constant c such that for all $r, n, m \in \mathbf{N}^+$ and $b = \lceil \log(n)^{cr^2 l^2} m^{crl} \rceil$, $G_m^n[v_{b+1}, \dots, v_n]$ contains a.a.s. no r-shallow topological minor of a graph with l vertices and more edges than vertices.*

Proof. If a graph contains an r-shallow topological minor of a graph with l vertices and more edges than vertices then it also contains a subgraph with at most $(2r+1)(l+1)$ vertices and more edges than vertices. Using this observation, this lemma follows directly from Corollary 13. ◀

▶ **Theorem 15.** *Let $r, n, m \in \mathbf{N}^+$. Every r-shallow topological clique minor in G_m^n has a.a.s. size at most $\log(n)^{O(r^2)} m^{O(r)}$.*

Proof. The previous lemma with $l = 4$ and $b = \lceil \log(n)^{16cr^2} m^{4cr} \rceil$ means that the graph $G_m^n[v_{b+1}, \dots, v_n]$ contains a.a.s. no r-shallow clique minor of size four. Since adding the vertices $v_1, \dots, v_b$ to the graph can increase the size of the maximum r-shallow topological clique minor by at most b, we immediately obtain this result. ◀

▶ **Lemma 16.** *There is a constant c such that for all $r, n, m \in \mathbf{N}^+$ and $b = \lceil \log(n)^{cr^2} m^{cr} \rceil$, a.a.s. every r-neighborhood of $G_m^n[v_{b+1}, \dots, v_n]$ is either a tree or a tree with one additional edge.*

Proof. Assume an r-neighborhood which is a tree with two or more additional edges. We build a breadth-first spanning tree T of the r-neighborhood with root v and radius r. We pick two edges a_1, a_2 and b_1, b_2 which are not in T. Now there are unique paths of length at most $2r+1$ in T between v and a_1, a_2, b_1, b_2. The two edges together with these four paths describe a graph with at most $4(2r+1)$ vertices and one more edge than vertices. The statement now follows from Corollary 13. ◀

▶ **Theorem 17.** *Let $m \in \mathbf{N}^+$ and $g(n)\colon \mathbf{N} \to \mathbf{N}$ be a function with $g(n) = \omega(1)$. If we remove the first $\log(n)^{g(n)}$ vertices from G_m^n the remaining graph has a.a.s. locally bounded treewidth.*

Proof. Let $G_m'^n$ be the graph obtained by removing the first $\log(n)^{g(n)}$ many vertices from G_m^n and let c be the constant from Lemma 16. For every $r \in \mathbf{N}$ there exists an $f(r)$ such that for $n \geq f(r)$, $G_m'^n$ was obtained by removing at least the first $\lceil \log(n)^{cr^2} m^{rc} \rceil$ vertices. Then by Lemma 16, a.a.s. every r-neighborhood of $G_m'^n$ is either a tree or a tree with an additional edge and therefore has treewidth at most 2. If $n \leq f(r)$ then trivially every r-neighborhood of $G_m'^n$ has treewidth at most $f(r)$. Thus, $G_m'^n$ a.a.s. has locally bounded treewidth. ◀

5 Conclusion

In our analysis of preferential attachment graphs we obtained a logarithmic lower bound on the size of shallow clique minors, that implies that preferential attachment graphs are a.a.s. somewhere dense. This implies that algorithmic techniques developed for nowhere dense graph classes are not directly applicable to preferential attachment graphs. This is complemented by matching polylogarithmic upper bounds. We further observed that the removal of a polylogarithmic number of vertices makes the graph locally extremely sparse. Real networks however, do not have this extremely sparse structure after removal of very few vertices. Our results therefore yield further evidence that the preferential attachment process alone is not sufficient to explain the structure of complex networks.

References

1 Réka Albert, Hawoong Jeong, and Albert-László Barabási. Internet: Diameter of the world-wide web. *Nature*, 401(6749):130, 1999.

2 Agnes Backhausz et al. Limit distribution of degrees in random family trees. *Electronic Communications in Probability*, 16:29–37, 2011.

3 Albert-László Barabási and Réka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

4 Hans L Bodlaender, Fedor V Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M Thilikos. (Meta) kernelization. *Journal of the ACM (JACM)*, 63(5):44, 2016.

5 Béla Bollobás and Oliver Riordan. The diameter of a scale-free random graph. *Combinatorica*, 24(1):5–34, 2004.

6 Béla Bollobás, Oliver Riordan, Joel Spencer, and Gábor Tusnády. The degree sequence of a scale-free random graph process. *Random Structures & Algorithms*, 18(3):279–290, May 2001.

7 Béla Bollobás and Oliver M Riordan. Mathematical results on scale-free random graphs. *Handbook of graphs and networks: from the genome to the internet*, pages 1–34, 2003.

8 Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proc. of the National Academy of Sciences*, 99(25):15879–15882, 2002.

9 Bruno Courcelle. Graph rewriting: An algebraic and logic approach. In *Formal Models and Semantics*, pages 193–242. Elsevier, 1990.

10 Erik D. Demaine, Felix Reidl, Peter Rossmanith, Fernando Sánchez Villaamil, Somnath Sikdar, and Blair D. Sullivan. Structural sparsity of complex networks: Bounded expansion in random models and real-world graphs. *J. Comput. Syst. Sci.*, 105:199–241, 2019. doi:10.1016/j.jcss.2019.05.004.

11 R. Diestel. *Graph Theory*. Springer, Heidelberg, 2010.

12 Sander Dommers, Remco van der Hofstad, and Gerard Hooghiemstra. Diameters in preferential attachment models. *Journal of Statistical Physics*, 139(1):72–107, 2010.

13 Jan Dreier and Peter Rossmanith. Motif counting in preferential attachment graphs. In *39th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2019, December 11-13, 2019, Bombay, India*, volume 150 of *LIPIcs*, pages 13:1–13:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.FSTTCS.2019.13.

14 K. Eickmeyer, A. C. Giannopoulou, S. Kreutzer, O-J. Kwon, M. Pilipczuk, R. Rabinovich, and S. Siebertz. Neighborhood complexity and kernelization for nowhere dense classes of graphs. In *44th ICALP*, volume 80 of *LIPIcs*, pages 63:1–63:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.63.

15 Matthew Farrell, Timothy D Goodrich, Nathan Lemons, Felix Reidl, Fernando Sánchez Villaamil, and Blair D Sullivan. Hyperbolicity, degeneracy, and expansion of random intersection graphs. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 29–41. Springer, 2015.

16 Martin Grohe, Stephan Kreutzer, and Sebastian Siebertz. Deciding First-Order Properties of Nowhere Dense Graphs. *JACM*, 64(3):17, 2017.

17 Martin Grohe and Nicole Schweikardt. First-order query evaluation with cardinality conditions. In *Proceedings of the 37th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, Houston, TX, USA, June 10-15, 2018*, pages 253–266. ACM, 2018. doi:10.1145/3196959.3196970.

18 Michał Karoński, Edward R Scheinerman, and Karen B Singer-Cohen. On random intersection graphs: The subgraph problem. *Combinatorics, Probability and Computing*, 8(1-2):131–159, 1999.

19 Jon Kleinberg. The Small-World Phenomenon: An Algorithmic Perspective. In *Proceedings of the 32nd Symposium on Theory of Computing*, pages 163–170, 2000.

20 Dmitri Krioukov, Fragkiskos Papadopoulos, Maksim Kitsak, Amin Vahdat, and Marián Boguná. Hyperbolic geometry of complex networks. *Physical Review E*, 82(3):036106, 2010.

21 Stanley Milgram. The small world problem. *Psychology Today*, 2(1):60–67, 1967.

22 Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. Network motifs: simple building blocks of complex networks. *Science*, 298(5594):824–827, 2002.

23 Michael Molloy and Bruce Reed. A critical point for random graphs with a given degree sequence. *Random Structures & Algorithms*, 6(2-3):161–180, 1995.

24 Tamás F Móri. The maximum degree of the Barabási–Albert random tree. *Combinatorics, Probability and Computing*, 14(3):339–348, 2005.

25 Jaroslav Nešetřil and Patrice Ossona de Mendez. *Sparsity*. Springer, 2012.

26 Jaroslav Nešetřil, Patrice Ossona de Mendez, and David R Wood. Characterisations and examples of graph classes with bounded expansion. *European Journal of Combinatorics*, 33(3):350–373, 2012.

27 J. Nešetřil and P. Ossona de Mendez. Grad and classes with bounded expansion I. Decompositions. *European Journal of Combinatorics*, 29(3):760–776, 2008.

28 Erol Peköz, Adrian Röllin, and Nathan Ross. Joint degree distributions of preferential attachment random graphs. *Advances in Applied Probability*, 49(2):368–387, 2017.

29 Erol A Peköz, Adrian Röllin, Nathan Ross, et al. Degree asymptotics with rates for preferential attachment random graphs. *The Annals of Applied Probability*, 23(3):1188–1218, 2013.

30 Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

31 Remco van der Hofstad. *Random graphs and complex networks*, volume 1. Cambridge University Press, 2016.

32 Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.

On Nonadaptive Security Reductions of Hitting Set Generators

Shuichi Hirahara
National Institute of Informatics, Tokyo, Japan
s_hirahara@nii.ac.jp

Osamu Watanabe
Tokyo Institute of Technology, Japan
watanabe@c.titech.ac.jp

Abstract

One of the central open questions in the theory of average-case complexity is to establish the equivalence between the worst-case and average-case complexity of the Polynomial-time Hierarchy (PH). One general approach is to show that there exists a PH-computable hitting set generator whose security is based on some NP-hard problem. We present the limits of such an approach, by showing that there exists no exponential-time-computable hitting set generator whose security can be proved by using a nonadaptive randomized polynomial-time reduction from any problem outside $\mathsf{AM} \cap \mathsf{coAM}$, which significantly improves the previous upper bound $\mathsf{BPP}^{\mathsf{NP}}$ of Gutfreund and Vadhan (RANDOM/APPROX 2008 [14]). In particular, any security proof of a hitting set generator based on some NP-hard problem must use either an adaptive or non-black-box reduction (unless the polynomial-time hierarchy collapses). To the best of our knowledge, this is the first result that shows limits of black-box reductions from an NP-hard problem to some form of a distributional problem in DistPH.

Based on our results, we argue that the recent worst-case to average-case reduction of Hirahara (FOCS 2018 [18]) is inherently non-black-box, without relying on any unproven assumptions. On the other hand, combining the non-black-box reduction with our simulation technique of black-box reductions, we exhibit the existence of a "non-black-box selector" for GapMCSP, i.e., an efficient algorithm that solves GapMCSP given as advice two circuits one of which is guaranteed to compute GapMCSP.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Pseudorandomness and derandomization

Keywords and phrases hitting set generator, black-box reduction, average-case complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.15

Category RANDOM

Related Version A full version of the paper is available at `https://eccc.weizmann.ac.il/report/2019/025/`.

Funding Shuichi Hirahara and Osamu Watanabe: JSPS KAKENHI Grant Number 18H04090 JP
Shuichi Hirahara: Supported by ACT-I, JST.

1 Introduction

The technique of reductions is one of central tools in complexity theory. In order to show that a computational task A is easier than another computational task B, it suffices to design a (black-box) reduction, i.e., the algorithm that solves A given oracle access to B. Most reductions of complexity theory are *black-box*. That is, the correctness of a reduction can be established without assuming any computational efficiency of the oracle. Black-box reductions are quite powerful and led us to, for instance, the discovery of thousands of NP-complete problems computationally equivalent to each other. However, a line of work


© Shuichi Hirahara and Osamu Watanabe;
licensed under Creative Commons License CC-BY



has exhibited limits of black-box reductions: Black-box reductions are too general to resolve several important open questions. We herein continue the study of black-box reductions especially in the context of the construction of a hitting set generator.

A *hitting set generator* γ-secure against a class $\mathcal{C}$ is a family of functions $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ such that no $\mathcal{C}$-algorithm can γ-avoid G; here we say that an algorithm R *γ-avoids* G if R rejects every string in the image of G, and R accepts at least a γ-fraction of all inputs of length ℓ for every $\ell \in \mathbb{N}$. By default, we assume $\gamma := 1/4$ and we say that R *avoids* G if R $(1/4)$-avoids G. A typical approach for constructing a hitting set generator is to design a black-box reduction that reduces some computationally hard task to the task of avoiding a hitting set generator.

In fact, there have been already several known proof techniques that are not black-box. Impagliazzo and Wigderson [27] constructed a hitting set generator based on the uniform hardness assumption that $\mathsf{EXP} \neq \mathsf{BPP}$. Their security proof of the hitting set generator is not a (black-box) reduction from the task of solving EXP to the task of avoiding the hitting set generator; they crucially exploited the fact that there exists an efficient algorithm that avoids the hitting set generator. Trevisan and Vadhan [36] and Gutfreund and Vadhan [14] showed that the security reduction of [27] is inherently non-black-box in some senses. More recently, building on [10, 22], Hirahara [18] applied the proof techniques for constructing a hitting set generator to the context of average-case complexity, and presented the first non-black-box worst-case to average-case reduction within NP.

Given the fact that there are already non-black-box proof techniques, why should we study the limits of black-box reductions? We highlight several points:

1. Black-box reductions are *more general* and *useful* than non-black-box reductions. Therefore, it is desirable to have a black-box reduction when it is possible; studying limits of black-box reductions enables us to identify when one can hope to construct a black-box reduction.
   For example, Impagliazzo and Wigderson [27] showed that $\mathsf{EXP} \not\subseteq \mathsf{BPP}$ implies that BPP can be derandomized in *sub-exponential* time (on most inputs, for infinitely many input lengths). This is shown by a non-black-box reduction, and it is not known whether the result can be generalized to a "high-end" result: does $\mathsf{EXP} \not\subseteq \mathsf{BPSUBEXP}$ imply that BPP can be derandomized in *quasi-polynomial* time? On one hand, Trevisan and Vadhan [36] used a black-box reduction and provided a positive answer to this question when EXP is replaced with PSPACE. On the other hand, Gutfreund and Vadhan [14] showed that a (mildly adaptive) black-box reduction cannot be used to prove the "high-end" result for EXP.
2. Studying limits of black-box reductions can inspire new black-box reductions. Inspired by this work, Hirahara [21, 20] subsequently presented new constructions of black-box reductions to Kolmogorov complexity, which were previously conjectured to be impossible.
3. Surprisingly, in some cases, the proof techniques for showing limits of black-box reductions can be combined with non-black-box reductions. We will show that one of our new algorithms for simulating black-box reductions can be combined with a non-black-box reduction of [18] (under some assumptions), and present a new structural property of an approximation version of the Minimum Circuit Size Problem (MCSP [28]).

As a main result of this paper, we show that any security proof of a hitting set generator based on some NP-hard problem must use either an adaptive or non-black-box reduction. This is the first limit of black-box worst-case to average-case reductions from NP-hard problems to some form of a distributional problem in DistPH.

Due to the connection to several research areas such as average-case complexity, black-box reductions, and derandomization, it is not easy to describe the literature in few words; we thus review the literature in the subsequent two sections. In Section 2, we review the theory of average-case complexity and state our main results. In Section 3, we review the non-black-box reduction of [18], present some applications of our results, and describe our proof techniques. Due to the space limitation, details are omitted in this version; see the full version of the paper.

2 Average-Case Complexity

2.1 Background

One of the central open questions in the theory of average-case complexity [29] is to establish the equivalence between the worst-case and average-case complexity of NP.

▶ **Open Question 1.** *Does* $\mathsf{DistNP} \subseteq \mathsf{AvgP}$ *imply* $\mathsf{NP} = \mathsf{P}$*?*

Here DistNP is the class of distributional problems $(L, \mathcal{D})$ (i.e., a pair of a problem and its input distribution) such that $L \in \mathsf{NP}$ and $\mathcal{D}$ is an efficiently samplable distribution. AvgP is the class of distributional problems that admit an errorless heuristic polynomial-time scheme [8] (also known as an "average-case polynomial-time algorithm"). Here, for $L \subseteq \{0,1\}^*$ and $\mathcal{D} = \{\mathcal{D}_m\}_{m \in \mathbb{N}}$, a distributional problem $(L, \mathcal{D})$ is said to be in AvgP if there exists an algorithm M such that, for every $m \in \mathbb{N}$, given an input x in the support of D_m, and a parameter $\delta > 0$,

1. $M(x, \delta)$ halts in time $\mathsf{poly}(m, 1/\delta)$,
2. $M(x, \delta)$ outputs either the correct answer $L(x)$ or $\perp$ ("I don't know"), and
3. the probability that $M(x, \delta)$ outputs $\perp$ over a choice of $x \sim \mathcal{D}_m$ is at most δ.

Open Question 1 is of particular importance from the perspective of cryptography: Average-case hardness of NP is a prerequisite for constructing secure complexity-theoretic cryptographic primitives such as one-way functions (OWFs). Thus resolving Open Question 1 is an important step towards building cryptographic primitives whose security is based on more plausible assumptions (e.g., the worst-case hardness of NP).

There has been a line of work showing that Open Question 1 cannot be resolved by using either relativizing proof techniques [26], black-box worst-case-to-average-case reductions [11, 9, 2, 7, 6], or error-correcting codes [37].

For large enough complexity classes such as PSPACE and EXP, there is a general technique for converting any worst-case hard function f to some two-sided-error average-case hard function $\mathrm{Enc}(f)$ based on error-correcting codes [35, 36]. Here, the encoded function $\mathrm{Enc}(f)$ is computable in EXP or PSPACE given oracle access to f; thus, the worst-case and average-case complexity of such large complexity classes are known to be equivalent. Viola [37] showed limits of such an approach: $\mathrm{Enc}(f)$ cannot be computed in PH^f; thus, the proof technique of using error-correcting codes is not sufficient to resolve Open Question 1 as well as the following weaker open question:

▶ **Open Question 2.** *Does* $\mathsf{DistPH} \subseteq \mathsf{AvgP}$ *imply* $\mathsf{PH} = \mathsf{P}$ *(or, equivalently,* $\mathsf{NP} = \mathsf{P}$*)?* [1]

[1] We mention in passing that Pavan, Santhanam, and Vinodchandran [32] made some progress, by proving that $\mathsf{DistP}^{\mathsf{NP}} \subseteq \mathsf{AvgP}$ implies $\mathsf{NP} = \mathsf{P}$, under the implausible assumption that $\mathsf{NP} \subseteq \mathsf{P/poly}$.

Note that Open Question 2 is an easier question than Open Question 1, since $\mathsf{PH} = \mathsf{P}$ is known to be equivalent to $\mathsf{NP} = \mathsf{P}$. In fact, this well-known equivalence between $\mathsf{PH} = \mathsf{P}$ and $\mathsf{NP} = \mathsf{P}$ is shown by using a non-black-box reduction technique[2]; as we will explain later, this is one reason why all the previous limits of black-box reductions [11, 9, 2, 7, 6] fail to explain the difficulty of resolving Open Question 2. In this work, we present the first limit of black-box reduction techniques for resolving Open Question 2, thereby clarifying what kind of proof techniques are useful. We emphasize that, while Viola's result [37] excludes the construction of error-correcting codes within PH, it does not show limits of black-box worst-case to average-case reduction techniques such as Ajtai's reduction [1].

One general approach for constructing an (errorless) average-case hard function is to make use of a hitting set generator. Indeed, a hitting set generator G secure against polynomial-time algorithms naturally induces a hard distributional problem in DistNP^G: Consider the distributional problem $(\mathrm{Im}(G), \mathcal{U})$, i.e., the distributional problem of checking whether an input x is in the image of G, where x is randomly chosen from the uniform distribution $\mathcal{U}$. Since the number of YES instances of $\mathrm{Im}(G)$ is small under the uniform distribution, any errorless heuristic algorithm must reject a large fraction of NO instances, which gives rise to an algorithm that avoids G. To summarize:

▶ **Fact 3** (Implicit in [18]). *Suppose there exists a hitting set generator $G := \{G_\ell \colon \{0,1\}^{\ell-1} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ that is $1/4$-secure against polynomial-time algorithms. Then, $\mathsf{DistNP}^G \not\subseteq \mathsf{AvgP}$. In particular, when G is computable in PH, we obtain $\mathsf{DistPH} \not\subseteq \mathsf{AvgP}$.*

Fact 3 suggests an approach for resolving Open Question 2: Try to construct a PH-computable hitting set generator whose security is based on the worst-case hardness of NP. How do we compare this approach with the technique based on error-correcting codes [37]? Our approach is *more general*, because, given a two-sided-error average-case hard function $\mathrm{Enc}(f)$, one can construct a pseudorandom generator $G = \{G_\ell \colon \{0,1\}^{\ell-1} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$ defined as $G_\ell(z) := (z, \mathrm{Enc}(f)(z))$ for a seed $z \in \{0,1\}^{\ell-1}$ [38].

In order to construct a secure hitting set generator based on the hardness of a problem L, we need to argue that, if there exists an efficient algorithm that avoids G, then L can be solved efficiently. A typical way to establish such an implication is to design *black-box reductions* from L to a distinguisher for a hitting set generator. Specifically, for a candidate hitting set generator G, a reduction M is said to be a black-box reduction from L to any γ-avoiding oracle R for G if, for every input x and any oracle R that γ-avoids G, M computes L on input x under the oracle R.

Gutfreund and Vadhan [14] initiated the study of limits of such a black-box reduction, motivated by the question on whether derandomization is possible under uniform assumptions (e.g., [27, 36]). They showed that any polynomial-time randomized nonadaptive black-box reductions to any oracle avoiding an exponential-time-computable hitting set generator G can be simulated in $\mathsf{BPP}^{\mathsf{NP}}$. Unfortunately, their upper bound is too weak to deduce any limit of the approach on Open Question 2 since $\mathsf{NP} \subseteq \mathsf{BPP}^{\mathsf{NP}}$. Similarly, it is impossible to deduce any limit of the approach on Open Question 1, because the upper bound becomes trivial when G is polynomial-time-computable.

[2] Indeed, if the reduction is black-box, we should have $\mathsf{PH} \subseteq \mathsf{P}^{\mathsf{NP}}$, which means that PH collapses.

2.2 Our Results: Limits of Security Proof of Hitting Set Generators

We significantly improve the upper bound of [14] to $\mathsf{AM} \cap \mathsf{coAM}$. We also show upper bounds of $\mathsf{NP/poly} \cap \mathsf{coNP/poly} \cap \mathsf{S}_2^{\mathsf{NP}}$ even if G is not computable.

To state our results formally, let $\mathsf{BPP}_{\|}^R$ denote the class of languages solvable by a randomized polynomial-time machine with nonadaptive oracle access to R.[3] In the definition of a black-box reduction M to any γ-avoiding oracle R, the reduction M is not allowed to depend on R. However, we will show that the existence of a randomized nonadaptive black-box reduction from L to any γ-avoding oracle R is equivalent to saying that $L \in \mathsf{BPP}_{\|}^R$ for every oracle R that γ-avoids R.[4] In light of this, the result of Gutfreund and Vadhan [14] can be stated as $\bigcap_R \mathsf{BPP}_{\|}^R \subseteq \mathsf{BPP}^{\mathsf{NP}}$, where the intersection is taken over all oracles R that γ-avoid an exponential-time computable function G. Our main result improves $\mathsf{BPP}^{\mathsf{NP}}$ to $\mathsf{AM} \cap \mathsf{coAM}$:

▶ **Theorem 4** (Main). *Let $G = \{G_\ell : \{0,1\}^{s(\ell)} \to \{0,1\}^\ell\}_{\ell \in \mathbb{N}}$ be any (not necessarily computable) family of functions and $\gamma\colon \mathbb{N} \to [0,1)$ be a parameter such that*

- *there exists a constant $\epsilon > 0$ such that $s(\ell) \leq (1-\epsilon)\ell$ for all large $\ell \in \mathbb{N}$, and*
- *there exists a constant $c > 0$ such that $\gamma(\ell) \leq 1 - \ell^{-c}$ for all large $\ell \in \mathbb{N}$.*

Then,

$$\bigcap_R \mathsf{BPP}_{\|}^R \quad \subseteq \quad \mathsf{NP/poly} \cap \mathsf{coNP/poly} \cap \mathsf{S}_2^{\mathsf{NP}},$$

where the intersection is taken over all oracles R that γ-avoids G. Moreover, if G can be computed in time $2^{O(\ell)}$, then we also have

$$\bigcap_R \mathsf{BPP}_{\|}^R \quad \subseteq \quad \mathsf{AM} \cap \mathsf{coAM}.$$

At the core of Theorem 4 is the following two types of algorithms simulating black-box reductions: One is an $\mathsf{S}_2^{\mathrm{p}}$-type algorithm that simulates any query $q \stackrel{?}{\in} R$ of length at most $\Theta(\log n)$, and the other is an $\mathsf{AM} \cap \mathsf{coAM}$-type algorithm that simulates any query $q \stackrel{?}{\in} R$ of length at least $\Theta(\log n)$. In particular, if G is exponential-time computable, the $\mathsf{S}_2^{\mathrm{p}}$-type algorithm can be replaced with a polynomial-time algorithm and obtain the $\mathsf{AM} \cap \mathsf{coAM}$ upper bound.

Theorem 4 shows that there exists no hitting set generator whose security can be based on the hardness of some NP-hard problem via a nonadaptive reduction (unless $\mathsf{NP} \subseteq \mathsf{coNP/poly}$). In particular, the approach for Open Question 2 by constructing a PH-computable hitting set generator based on an NP-hard problem must use either an adaptive or non-black-box reduction.

It is worthy of note that Theorem 4 is almost tight from several perspectives: First, it is impossible to extend Theorem 4 to the case of *adaptive* reductions (unless $\mathsf{PSPACE} = \mathsf{AM}$). Indeed, Trevisan and Vadhan [36] constructed an exponential-time-computable pseudorandom generator based on the intractability of some PSPACE-complete problem, and its security reduction is adaptive and black-box in the sense of Theorem 4. Second, our $\mathsf{S}_2^{\mathrm{p}}$-type algorithm for simulating short queries is completely tight when G is a universal Turing machine. Third, it is possible to construct a hitting set generator based on the hardness of SZK (Statistical

[3] The subscript $\|$ stands for parallel queries.

[4] We state our results in the latter way because this makes our impossibility results stronger. The proof of the equivalence can be found in the full version of the paper.

Zero Knowledge), which is one of the best lower bound on $\mathsf{AM} \cap \mathsf{coAM}$; thus, the $\mathsf{AM} \cap \mathsf{coAM}$ upper bound of Theorem 4 cannot be significantly improved. (The details can be found in the full version of the paper.)

2.3 Related Work: Limits of Worst-case to Average-case Reductions within NP

To the best of our knowledge, Theorem 4 is *the first result* that shows limits of black-box reductions from an NP-hard problem to (some form of) a distributional problem in DistPH. In order to explain this in more detail, we review the previous work on limits of worst-case to average-case reductions within NP.

A natural approach for establishing the equivalence between the worst-case and average-case complexity of NP is by means of black-box reductions. That is, it is sufficient for resolving Open Question 1 to design a reduction that solves some NP-hard problem L, using oracle access to an errorless heuristic algorithm M that solves some distributional problem in DistNP. A line of work has been devoted to explaining why such a black-box reduction technique is too general to establish a worst-case to average-case connection for an NP-complete problem.

Building on the work of Feigenbaum and Fortnow [11], Bogdanov and Trevisan [9] showed that if a worst-case problem L is reducible to some distributional problem in DistNP via a nonadaptive black-box randomized polynomial-time reduction, then L must be in $\mathsf{NP/poly} \cap \mathsf{coNP/poly}$. This in particular shows that the average-case hardness of NP cannot be based on the worst-case hardness of an NP-complete problem using such a reduction technique (unless the polynomial-time hierarchy collapses [39]). Akavia, Goldreich, Goldwasser and Moshkovitz [2, 3] showed that, in the special case of a nonadaptive reduction to the task of inverting a one-way function, the upper bound of [9] can be improved to $\mathsf{AM} \cap \mathsf{coAM}$, thereby removing the advice "/poly". Bogdanov and Brzuska [7] showed that even an adaptive reduction to the task of inverting a size-verifiable one-way function cannot be used for any problem outside $\mathsf{AM} \cap \mathsf{coAM}$. Applebaum, Barak, and Xiao [6] studied black-box reductions to PAC learning, and observed that the technique of [2] can be applied to (some restricted type of) a black-box reduction to the task of inverting an auxiliary-input one-way function (AIOWF), which is a weaker primitive than a one-way function. We summarize the limits of black-box reductions (depicted by $\rightarrow$) as well as known implications (depicted by $\Longrightarrow$) in Figure 1.

Compared to the previous results on the limits of black-box worst-case-to-average-case reductions within NP, a surprising aspect of Theorem 4 is that it generalizes to any function G that may not be computable (and this is a key property for obtaining the limits of the approach on Open Question 2). Indeed, almost all the previous results [11, 9, 2, 6] crucially exploit the fact that a verifier can check the correctness of a certificate for an NP problem; thus a dishonest prover can cheat the verifier only in one direction by not providing a certificate for a YES instance. In our simulation algorithms, a verifier cannot compute G and thus cannot prevent dishonest provers from cheating in this way. At a high level, our technical contributions are to overcome this difficulty by combining the ideas of Gutfreund and Vadhan [14] with the techniques developed in [11, 9].

Is it possible to directly deduce some limits of an approach on Open Question 2 from the previous results [11, 9]? No! Recall that, in order to resolve Open Question 2, it suffices to establish a reduction from an NP-complete problem to DistPH (using the non-black-box equivalence between $\mathsf{P} = \mathsf{NP}$ and $\mathsf{P} = \mathsf{PH}$). The results of [11, 9] crucially rely on the fact that a YES instance of DistNP is verifiable in polynomial time. If we would like to simulate

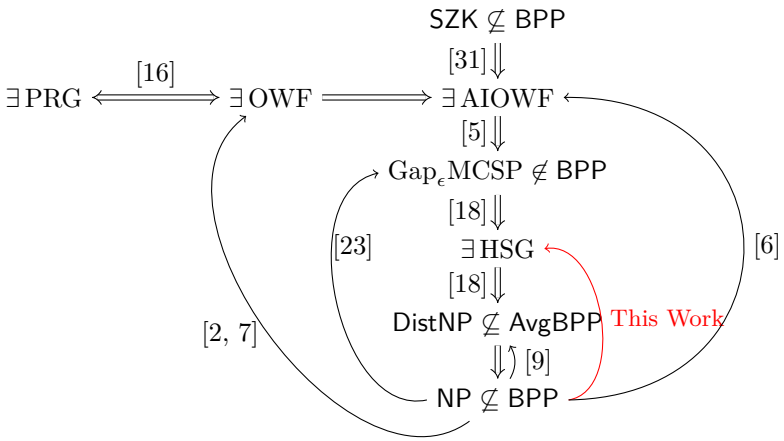


Figure 1 Average-case complexity and limits of black-box reductions. "$A \to B$" means that there is no black-box (or oracle-independent) reduction technique showing "$A \Rightarrow B$" under reasonable complexity theoretic assumptions. The security of all cryptographic primitives is with respect to an almost-everywhere polynomial-time randomized adversary.

a black-box reduction to DistNP^A for some oracle A, the simulation protocol of Feigenbaum and Fortnow [11] runs in $\mathsf{NP}^A/\mathsf{poly} \cap \mathsf{coNP}^A/\mathsf{poly}$. Thus, in order to simulate a reduction to $\mathsf{Dist}\Sigma_2^{\mathrm{p}} \subseteq \mathsf{DistPH}$, the upper bound becomes $\mathsf{NP}^{\mathsf{NP}}/\mathsf{poly} \cap \mathsf{coNP}^{\mathsf{NP}}/\mathsf{poly}$, which trivially contains NP.

It is also worthy of note that Theorem 4 improves some aspects of all the previous results about limits of black-box reductions within NP. Compared to [9], our results show that the advice "/poly" is not required to simulate black-box reductions to any oracle avoiding an exponential-time-computable hitting set generator. Compared to [2, 6], our results are "improvement" on their results in the sense that the existence of auxiliary-input one-way functions implies the existence of hitting set generators; on the other hand, since the implication goes through the *adaptive* reduction (from the task of inverting a one-way function to a distinguisher for a PRG) of [16], technically speaking, our results are incomparable with their results.[5] Similarly, our results conceptually improve the result of [23], but these are technically incomparable, mainly because the implication goes through the non-black-box reduction of [18].

3 In Search of Inherently Non-Black-Box Reduction Techniques

Hirahara [18] presented the first non-black-box worst-case to average-case reduction within NP, which is the motivation for this work. Building on [10, 22], Hirahara [18] presented a (nonadaptive) reduction from Gap$_\epsilon$MCSP to a distinguisher for a polynomial-time-computable hitting set generator $G^{\mathsf{int}} = \{G^{\mathsf{int}}_{2^n} \colon \{0,1\}^{\widetilde{O}(2^{\delta n})} \to \{0,1\}^{2^n}\}_{n \in \mathbb{N}}$. Here, G^{int} is a "circuit

[5] We emphasize that we concern the nonadaptivity of reductions used in the security proof of pseudorandom generators. Several simplified constructions of pseudorandom generators G^f from one-way functions f (e.g., [25, 15]) are nonadaptive in the sense that G^f can be efficiently computed with nonadaptive oracle access to f; however, the security reductions of these constructions are adaptive because of the use of Holenstein's uniform hardcore lemma [24]. Similarly, the reduction of [16, Lemma 6.5] is adaptive. (We note that, in the special case when the degeneracy of a one-way function is efficiently computable, the reduction of [16] is nonadaptive.)

interpreter": a function that takes a description of a circuit of size $2^{\delta n}$ and outputs its truth table. For a constant $\epsilon > 0$, $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ denotes the problem of approximating the minimum circuit size of a Boolean function $f\colon \{0,1\}^n \to \{0,1\}$ within a factor of $2^{(1-\epsilon)n}$, given the truth table of f. Rudich [33] conjectured that $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ cannot be solved in $\mathsf{coNP/poly}$ (even in the sense of average-case complexity). Therefore, the reduction of [18] is indeed non-black-box under Rudich's conjecture, as otherwise it contradicts the limits of black-box reductions (such as Theorem 4 and [9]).

Here we pose the following question:

> Are the reductions of [18] *inherently* non-black-box? Or should we regard it as an approach for refuting Rudich's conjecture?

On one hand, the proofs of [18] seem to yield only non-black-box reductions, in the sense that the efficiency of an oracle is crucially exploited. On the other hand, if the reduction from $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ to DistNP could be made black-box, by using our coAM simulation protocol of black-box reductions (i.e., Theorem 4), we would obtain $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{coAM} \subseteq \mathsf{coNP/poly}$, which refutes Rudich's conjecture.

In order to answer the question, it is desirable to clarify a fundamental obstacle to applying the simulation techniques of black-box reductions to the reductions of [18], without relying on any unproven assumption.

3.1 Hirahara's Reduction is Unconditionally Non-Black-Box

Based on Theorem 4, we can argue that the reductions of [18] are inherently non-black-box in a certain formal sense. The reason is that the idea of [18] can be applied to not only time-bounded Kolmogorov complexity but also any other types of Kolmogorov complexity, including *resource-unbounded* Kolmogorov complexity. Therefore, if this generalized reduction could be made black-box, then (as outlined below) by Theorem 4 we would obtain a finite-running-time algorithm $\mathsf{S}_2^{\mathsf{NP}}$ that approximates resource-unbounded Kolmogorov complexity, which is a contradiction *unconditionally*.

More specifically, fix any universal Turing machine U, and regard it as a hitting set generator $U = \{U_\ell : \{0,1\}^{\ell/2} \to \{0,1\}^\ell\}_{\ell\in\mathbb{N}}$. That is, U_ℓ takes an input (M, x) of length $\ell/2$, simulates the Turing machine M on input x, and outputs $M(x)$ if the length of the output $M(x)$ is exactly ℓ; otherwise, U_ℓ outputs 1^ℓ.

▷ Claim 5. Suppose that there exists a computable oracle R that avoids U. Then, there exists a randomized polynomial-time nonadaptive R-oracle algorithm that approximates $\mathrm{K}_U(x)$.

Proof Sketch. The idea of the non-black-box reduction of [18] is as follows: Given an input $x \in \{0,1\}^n$, take any construction of a pseudorandom generator $G^x\colon \{0,1\}^{\ell/4} \to \{0,1\}^\ell$ based on a worst-case hard function $x\colon \{0,1\}^{\log n} \to \{0,1\}$.[6] For example, we can use the Nisan-Wigderson generator [30] combined with some error-correcting codes. The reduction estimates $p := \mathbb{E}_z[R(G^x(z))]$ by sampling, and accepts if and only if p is small.

The correctness is proved as follows: If $\mathrm{K}_U(x) \le \ell/5$, then $\mathrm{K}_U(G^x(z)) \le |z| + \mathrm{K}_U(x) \ll \ell/2$ for a large enough ℓ; thus $p = 0$. Conversely, if $p \approx 0$, then by using the security proof of G^x, we obtain a small R-oracle Turing machine that outputs x; thus $\mathrm{K}^R_U(x) \le \mathsf{poly}(\ell, \log n)$; in particular, *by using the assumption that R is computable*, we obtain $\mathrm{K}_U(x) \le \mathsf{poly}(\ell, \log n)$. Therefore, the reduction distinguishes the YES instances x such that $\mathrm{K}_U(x) \le \ell/5$ and the NO instances x such that $\mathrm{K}_U(x) > \mathsf{poly}(\ell, \log n)$. ◁

[6] Here we identify a function with its truth table.

Observe that, in the proof above, we crucially used the assumption that R is computable. Can we avoid the assumption and generalize Claim 5 for any R that avoids U? In other words, is there a black-box reduction from approximating $\mathrm{K}_U(x)$ to the task of avoiding U? If it is the case, Theorem 4 implies that approximating $\mathrm{K}_U(x)$ can be done in $\mathsf{S}_2^{\mathsf{NP}}$, which contradicts the undecidability of Kolmogorov complexity. Therefore, we conclude that the reduction of Claim 5 is inherently non-black-box.

3.2 Applications: Non-Black-Box Selector for GapMCSP

As explained in the previous subsection, the non-black-box reductions of [18] cannot be combined with Theorem 4 *unconditionally*. However, we show that our simulation protocol of black-box reductions can be combined with the non-black-box reductions *conditionally*, which constitutes a new structural property of GapMCSP – the existence of a "non-black-box selector."

▶ **Theorem 6** (GapMCSP Has a "Non-Black-Box Selector"). *For any constant $\epsilon > 0$, there exist some constant $\delta > 0$ and a randomized polynomial-time algorithm that takes as advice two circuits one of which is guaranteed to solve* $\mathrm{Gap}_\epsilon\mathrm{MCSP}$ *and solves* $\mathrm{Gap}_\delta\mathrm{MCSP}$ *with high probability.*

A *selector* for a problem L is an efficient algorithm that solves L given oracle access to two oracles one of which is guaranteed to solve; thus, it "selects" the correct answer from the two oracles. The notion of selector exactly characterizes the class of languages for which advice of logarithmic length can be removed [17]. The selector of Theorem 6 is non-black-box in the sense that it requires to take as advice two polynomial-size circuits instead of black-box access to two oracles.

The main building block of the non-black-box selector is our $\mathsf{S}_2^{\mathrm{p}}$-type simulation algorithm of Theorem 4. Recall that $\mathsf{S}_2^{\mathrm{p}}$ is a proof system where two competing provers, one of which is guaranteed to be honest, try to convince a polynomial-time verifier. In our $\mathsf{S}_2^{\mathrm{p}}$ simulation algorithm of black-box reductions, for each $i \in \{0,1\}$, the ith prover sends a set R_i; the honest prover sends a set R_i that avoids a hitting set generator G. Then a verifier obtains an oracle $R_0 \cap R_1$ that avoids G, to which the reduction is guaranteed to work.

Theorem 6 is proved by combining this $\mathsf{S}_2^{\mathrm{p}}$-type simulation algorithm with the non-black-box reductions of [10, 18].[7] The reason why we can combine the non-black-box reductions with our $\mathsf{S}_2^{\mathrm{p}}$-type simulation algorithm is that the non-black-box reduction of [18] is, in fact, a *size-restricted black-box reduction* [14]. This is a black-box reduction which works correctly when an oracle can be computed by a polynomial-size circuit. Our $\mathsf{S}_2^{\mathrm{p}}$-type simulation algorithm can simulate the size-restricted black-box reduction under the assumption that there exists a polynomial-size circuit that avoids a hitting set generator.

In contrast, we were not able to combine our $\mathsf{AM} \cap \mathsf{coAM}$ algorithm of Theorem 4 with the non-black-box reductions under similar conditions. We leave it as an interesting open question, which could have an application to fixed-polynomial circuit lower bounds (e.g., [34]).

▶ **Open Question 7** ("Non-Black-Box Instance Checkability" of GapMCSP). *Prove that* $\mathrm{MCSP} \in \mathsf{P/poly}$ *(or* $\mathsf{NP} \subseteq \mathsf{P/poly}$*) implies* $\mathrm{Gap}_\epsilon\mathrm{MCSP} \in \mathsf{coAM}$ *for some constant $\epsilon > 0$.*

[7] There is an alternative proof based on the search-to-decision reduction of GapMCSP given by Carmosino, Impagliazzo, Kabanets, and Kolokolova [10]. However, we choose to present the proof by combining the $\mathsf{S}_2^{\mathrm{p}}$-type simulation algorithm with the non-black-box reductions in order to highlight the difference between Theorem 6 and Open Question 7.

3.3 Our Techniques

We outline our proof strategy for Theorem 4 below. Suppose that we have some reduction M from L to any oracle R that avoids a hitting set generator G. Fix any input $x \in \{0,1\}^*$, and let $\mathcal{Q}_x$ denote the query distribution that a reduction makes on input x. We focus on the case when the length of each query is larger than $\Theta(\log n)$, and explain the proof ideas for showing $L \in \mathsf{AM} \cap \mathsf{coAM}$.

As a warm-up, consider the case when the support $\mathsf{supp}(\mathcal{Q}_x)$ of $\mathcal{Q}_x$ is small (i.e., $|\mathsf{supp}(\mathcal{Q}_x) \cap \{0,1\}^\ell| \ll 2^\ell$ for all large $\ell \in \mathbb{N}$). In this case, we can define an oracle R_1 so that $R_1 := \{0,1\}^* \setminus \mathsf{supp}(\mathcal{Q}_x) \setminus \mathrm{Im}(G)$; this avoids the hitting generator G because $R_1 \cap \mathrm{Im}(G) = \varnothing$ and the size of $R_1 \cap \{0,1\}^\ell$ is at least $2^\ell - |\mathsf{supp}(\mathcal{Q}_x)| - |\mathrm{Im}(G_\ell)| \gg 2^{\ell-1}$ for all large $\ell \in \mathbb{N}$. Therefore, it is guaranteed that the reduction M computes L correctly under the oracle R_1; we can simulate the reduction by simply answering all the queries by saying "No" (since $q \notin R_1$ for every $q \in \mathcal{Q}_x$); hence $L \in \mathsf{BPP}$.

In general, we cannot hope that $\mathsf{supp}(\mathcal{Q}_x)$ is small enough. To generalize the observation above, let us recall the notion of α-heaviness [9]: We say that a query q is *α-heavy* (with respect to $\mathcal{Q}_x$) if the query q is α times more likely to be sampled under $\mathcal{Q}_x$ than the uniform distribution on $\{0,1\}^{|q|}$; that is, $\Pr_{w \sim \mathcal{Q}_x}[w = q] \geq \alpha 2^{-|q|}$. Now we define our new oracle $R_2 := \{0,1\}^* \setminus \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\} \setminus \mathrm{Im}(G)$, which can again be shown to avoid G because the fraction of α-heavy queries is at most $1/\alpha$ $(\ll 1)$.

The problem now is that it is difficult to simulate the new oracle R_2; it appears that, given a query q, we need to test whether $q \stackrel{?}{\in} \mathrm{Im}(G)$, which is not possible in $\mathsf{AM} \cap \mathsf{coAM}$. However, it turns out that it is not necessary to test it, as we explain next: Observe that the size of $\mathrm{Im}(G)$ is very small; it is at most $2^{s(\ell)}$ $\left(\ll 2^\ell\right)$. Thus, the probability that a query q is in $\mathrm{Im}(G)$ and q is not α-heavy (i.e., q is rarely queried) is at most $\alpha \cdot 2^{s(\ell)-\ell}$, where ℓ denotes the length of q. As a consequence, the reduction cannot "distinguish" the oracle R_2 and a new oracle $R_3 := \{0,1\}^* \setminus \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\}$; hence, we can simulate the reduction if, given a query q, we can decide whether $q \stackrel{?}{\in} R_3$ in $\mathsf{AM} \cap \mathsf{coAM}$.

This task, however, still appears to be difficult for $\mathsf{AM} \cap \mathsf{coAM}$; indeed, at this point, Gutfreund and Vadhan [14] used the fact that the approximate counting is possible in $\mathsf{BPP}^{\mathsf{NP}}$, and thereby simulated the oracle R_3 by a $\mathsf{BPP}^{\mathsf{NP}}$ algorithm.

Our main technical contribution is to develop a way of simulating the reduction to R_3. First, note that the lower bound protocol of Goldwasser and Sipser [13] enables us to give an AM certificate for α-heaviness; we can check, given a query q, whether q is $\alpha(1+\epsilon)$-heavy or α-light for any small error parameter $\epsilon > 0$. Thus, we have an AM protocol for $\{0,1\}^* \setminus R_3$ for every query q (except for the queries that are α-heavy and $\alpha(1+\epsilon)$-light).

If, in addition, we had an AM protocol for R_3, then we would be done; however, it does not seem possible in general. The upper bound protocol of Fortnow [12] performs a similar task, but the protocol can be applied only for a limited purpose: we need to keep the randomness used to generate a query $q \sim \mathcal{Q}_x$ from being revealed to the prover. When the number of queries of the reduction is limited to 1, we can use the upper bound protocol in order to give an AM certificate for R_3; on the other hand, if the reduction makes two queries $(q_1, q_2) \sim \mathcal{Q}_x$, we cannot simultaneously provide AM certificates of the upper bound protocol for *both* q_1 and q_2, because the fact that q_1 and q_2 are sampled *together* may reveal some information about the private randomness. To summarize, the upper bound protocol works only for the *marginal* distribution of each query, but does not work for the *joint* distribution of several queries.

Still, the upper bound protocol is useful for extracting some information about *each* query. For example, the heavy-sample protocol of Bogdanov and Trevisan [9] (which combines the lower and upper bound protocol and sampling) estimates, in $\mathsf{AM} \cap \mathsf{coAM}$, the probability that a query q sampled from $\mathcal{Q}_x$ is α-heavy. This protocol enables us to estimate the probability that $q \in R_3$ over the choice of $q \sim \mathcal{Q}_x$.

The probability that $q \in R_3$ is useful for simulating the reduction M. Feigenbaum and Fortnow [11] developed an $\mathsf{AM} \cap \mathsf{coAM}$ protocol that simulates a nonadaptive reduction to an NP oracle R, given as advice the probability that a query q is in R. We generalize this protocol for the case when the oracle R is solvable by AM on average:

▶ **Theorem 8** (Generalized Feigenbaum–Fortnow Protocol; informal)**.** *Suppose that M is a randomized polynomial-time nonadaptive reduction to an oracle R whose queries are distributed according to $\mathcal{Q}_x$ on input $x \in \{0,1\}^n$, and that R is solvable by AM on average (i.e., there exists an AM protocol Π_R such that, with probability $1 - 1/\mathsf{poly}(n)$ over the choice of $q \sim \mathcal{Q}_x$, the protocol Π_R computes R on input q). Then, there exists an $\mathsf{AM} \cap \mathsf{coAM}$ protocol Π_M such that, given a probability $p^* \approx \Pr_{q \sim \mathcal{Q}_x}[q \in R]$ as advice, the protocol Π_M simulates the reduction M with probability at least $1 - 1/\mathsf{poly}(n)$.*

Let R denote the complement of R_3, i.e., $R := \{\, q \in \{0,1\}^* \mid q\colon \alpha\text{-heavy} \,\}$. Using the generalized Feigenbaum–Fortnow protocol, we simulate the reduction M to R as follows. Firstly, we use the heavy-sample protocol of [9] in order to estimate $p^* \approx \Pr_{q \sim \mathcal{Q}_x}[q\colon \alpha\text{-heavy}]$. Secondly, using the lower bound protocol of [13], we argue that R can be solved by some AM-protocol Π_R on average. Lastly, we use the protocol of Theorem 8 to simulate M. The details can be found in the full version of the paper.

We mention in passing the difficulty of Open Question 7, i.e., the reason why we were not able to combine our $\mathsf{AM} \cap \mathsf{coAM}$-type simulation algorithm with the non-black-box reduction *even conditionally*: The non-black-box reduction outlined in Subsection 3.1 reduces the promise problem whose YES instance consists of $\mathrm{K}_U(x) \leq \ell/5$ and NO instance consists of $\mathrm{K}_U^R(x) > \mathsf{poly}(\ell, \log n)$ to an oracle R. In order to make sure that the promise problem is non-trivial, it is important that R does not depend on x. On the other hand, in our simulation algorithm, we need to choose an oracle R_x depending on the input x, which potentially makes the promise problem trivial. (For example, $\mathrm{K}_U^{R_x}(x)$ may be always close to 0.)

3.4 Subsequent Work

Inspired by this work, Allender's conjectures [4] were refuted under the plausible assumptions about the exponential-time hierarchy [21, 20]. Moreover, it turned out that the stretch of a hitting set generator construction is important. In [20], it was shown that there exists a function $G = \{G : \{0,1\}^{n - O(\log n)} \to \{0,1\}^n\}_{n \in \mathbb{N}}$ such that $\mathsf{NEXP} \cup \mathsf{coNEXP} \subseteq \mathsf{BPP}_{\|}^R$ for any oracle R that avoids G. This result bypasses our limits of black-box reductions (Theorem 4) because G extends its seed by a small amount of $O(\log n)$ whereas Theorem 4 requires that G extends its seed by a constant factor. In [19], the approximation quality of non-black-box reductions of [18] is improved. Moreover, based on the improvement, it is shown that, under the assumption that $\mathsf{DistPH} \subseteq \mathsf{AvgP}$, the time-bounded SAT-oracle Kolmogorov complexity of a string x is equal to the time-bounded Kolmogorov complexity of x up to an additive term of $O(\log n)$, for any string $x \in \{0,1\}^n$.

References

1 Miklós Ajtai. Generating Hard Instances of Lattice Problems (Extended Abstract). In *Proceedings of the Symposium on the Theory of Computing (STOC)*, pages 99–108, 1996. doi:10.1145/237814.237838.

2 Adi Akavia, Oded Goldreich, Shafi Goldwasser, and Dana Moshkovitz. On basing one-way functions on NP-hardness. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 701–710, 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 *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 795–796, 2010. doi:10.1145/1806689.1806797.

4 Eric Allender. Curiouser and Curiouser: The Link between Incompressibility and Complexity. In *Proceedings of the 8th Conference on Computability in Europe (CiE)*, pages 11–16, 2012. doi:10.1007/978-3-642-30870-3_2.

5 Eric Allender and Shuichi Hirahara. New Insights on the (Non-)Hardness of Circuit Minimization and Related Problems. In *Proceedings of the International Symposium on Mathematical Foundations of Computer Science (MFCS)*, pages 54:1–54:14, 2017. doi:10.4230/LIPIcs.MFCS.2017.54.

6 Benny Applebaum, Boaz Barak, and David Xiao. On Basing Lower-Bounds for Learning on Worst-Case Assumptions. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 211–220, 2008. doi:10.1109/FOCS.2008.35.

7 Andrej Bogdanov and Christina Brzuska. On Basing Size-Verifiable One-Way Functions on NP-Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 1–6, 2015. doi:10.1007/978-3-662-46494-6_1.

8 Andrej Bogdanov and Luca Trevisan. Average-Case Complexity. *Foundations and Trends in Theoretical Computer Science*, 2(1), 2006. doi:10.1561/0400000004.

9 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.

10 Marco L. Carmosino, Russell Impagliazzo, Valentine Kabanets, and Antonina Kolokolova. Learning Algorithms from Natural Proofs. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 10:1–10:24, 2016. doi:10.4230/LIPIcs.CCC.2016.10.

11 Joan Feigenbaum and Lance Fortnow. Random-Self-Reducibility of Complete Sets. *SIAM J. Comput.*, 22(5):994–1005, 1993. doi:10.1137/0222061.

12 Lance Fortnow. The Complexity of Perfect Zero-Knowledge. *Advances in Computing Research*, 5:327–343, 1989.

13 Shafi Goldwasser and Michael Sipser. Private Coins versus Public Coins in Interactive Proof Systems. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 59–68, 1986. doi:10.1145/12130.12137.

14 Dan Gutfreund and Salil P. Vadhan. Limitations of Hardness vs. Randomness under Uniform Reductions. In *Proceedings of the Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM)*, pages 469–482, 2008. doi:10.1007/978-3-540-85363-3_37.

15 Iftach Haitner, Omer Reingold, and Salil P. Vadhan. Efficiency Improvements in Constructing Pseudorandom Generators from One-Way Functions. *SIAM J. Comput.*, 42(3):1405–1430, 2013. doi:10.1137/100814421.

16 Johan Håstad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A Pseudorandom Generator from any One-way Function. *SIAM J. Comput.*, 28(4):1364–1396, 1999. doi:10.1137/S0097539793244708.

17 Shuichi Hirahara. Identifying an Honest $\mathsf{EXP}^{\mathsf{NP}}$ Oracle Among Many. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 244–263, 2015. doi:10.4230/LIPIcs.CCC.2015.244.

18 Shuichi Hirahara. Non-black-box Worst-case to Average-case Reductions within NP. In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 247–258, 2018.

19 Shuichi Hirahara. Non-Disjoint Promise Problems from Meta-Computational View of Pseudorandom Generator Constructions. To appear in CCC'20, 2020.

20 Shuichi Hirahara. Unexpected Hardness Results for Kolmogorov Complexity Under Uniform Reductions. *To appear in STOC'20*, 2020.

21 Shuichi Hirahara. Unexpected Power of Random Strings. In *Proceedings of the Innovations in Theoretical Computer Science Conference (ITCS)*, pages 41:1–41:13, 2020. doi:10.4230/LIPIcs.ITCS.2020.41.

22 Shuichi Hirahara and Rahul Santhanam. On the Average-Case Complexity of MCSP and Its Variants. In *Proceedings of the Computational Complexity Conference (CCC)*, pages 7:1–7:20, 2017. doi:10.4230/LIPIcs.CCC.2017.7.

23 Shuichi Hirahara and Osamu Watanabe. Limits of Minimum Circuit Size Problem as Oracle. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 18:1–18:20, 2016. doi:10.4230/LIPIcs.CCC.2016.18.

24 Thomas Holenstein. Key agreement from weak bit agreement. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 664–673, 2005. doi:10.1145/1060590.1060689.

25 Thomas Holenstein. Pseudorandom Generators from One-Way Functions: A Simple Construction for Any Hardness. In *Proceedings of the Theory of Cryptography Conference (TCC)*, pages 443–461, 2006. doi:10.1007/11681878_23.

26 Russell Impagliazzo. Relativized Separations of Worst-Case and Average-Case Complexities for NP. In *Proceedings of the Conference on Computational Complexity (CCC)*, pages 104–114, 2011. doi:10.1109/CCC.2011.34.

27 Russell Impagliazzo and Avi Wigderson. Randomness vs Time: Derandomization under a Uniform Assumption. *J. Comput. Syst. Sci.*, 63(4):672–688, 2001. doi:10.1006/jcss.2001.1780.

28 Valentine Kabanets and Jin-yi Cai. Circuit minimization problem. In *Proceedings of the Symposium on Theory of Computing (STOC)*, pages 73–79, 2000. doi:10.1145/335305.335314.

29 Leonid A. Levin. Average Case Complete Problems. *SIAM J. Comput.*, 15(1):285–286, 1986. doi:10.1137/0215020.

30 Noam Nisan and Avi Wigderson. Hardness vs Randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, 1994. doi:10.1016/S0022-0000(05)80043-1.

31 Rafail Ostrovsky. One-Way Functions, Hard on Average Problems, and Statistical Zero-Knowledge Proofs. In *Proceedings of the Structure in Complexity Theory Conference*, pages 133–138, 1991. doi:10.1109/SCT.1991.160253.

32 Aduri Pavan, Rahul Santhanam, and N. V. Vinodchandran. Some Results on Average-Case Hardness Within the Polynomial Hierarchy. In *Proceedings of the Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 188–199, 2006. doi:10.1007/11944836_19.

33 Steven Rudich. Super-bits, Demi-bits, and NP/qpoly-natural Proofs. In *Proceedings of the Randomization and Approximation Techniques in Computer Science (RANDOM/APPROX)*, pages 85–93, 1997. doi:10.1007/3-540-63248-4_8.

34 Rahul Santhanam. Circuit Lower Bounds for Merlin–Arthur Classes. *SIAM J. Comput.*, 39(3):1038–1061, 2009. doi:10.1137/070702680.

35 Madhu Sudan, Luca Trevisan, and Salil P. Vadhan. Pseudorandom Generators without the XOR Lemma. *J. Comput. Syst. Sci.*, 62(2):236–266, 2001. doi:10.1006/jcss.2000.1730.

36 Luca Trevisan and Salil P. Vadhan. Pseudorandomness and Average-Case Complexity Via Uniform Reductions. *Computational Complexity*, 16(4):331–364, 2007. doi:10.1007/s00037-007-0233-x.

37 Emanuele Viola. The complexity of constructing pseudorandom generators from hard functions. *Computational Complexity*, 13(3-4):147–188, 2005. doi:10.1007/s00037-004-0187-1.

38 Andrew Chi-Chih Yao. Theory and Applications of Trapdoor Functions (Extended Abstract). In *Proceedings of the Symposium on Foundations of Computer Science (FOCS)*, pages 80–91, 1982. doi:10.1109/SFCS.1982.45.

39 Chee-Keng Yap. Some Consequences of Non-Uniform Conditions on Uniform Classes. *Theor. Comput. Sci.*, 26:287–300, 1983. doi:10.1016/0304-3975(83)90020-8.

Testable Properties in General Graphs and Random Order Streaming

Artur Czumaj
Department of Computer Science and Centre for Discrete Mathematics and its Applications (DIMAP), University of Warwick, Coventry, UK
A.Czumaj@warwick.ac.uk

Hendrik Fichtenberger
Department of Computer Science, TU Dortmund, Germany
hendrik.fichtenberger@tu-dortmund.de

Pan Peng
Department of Computer Science, University of Sheffield, UK
p.peng@sheffield.ac.uk

Christian Sohler
Department of Mathematics and Computer Science, University of Cologne, Germany
csohler@uni-koeln.de

Abstract

We consider the fundamental question of understanding the relative power of two important computational models: *property testing* and *data streaming*. We present a novel framework closely linking these areas in the setting of *general graphs* in the context of constant-query complexity testing and constant-space streaming. Our main result is a generic *transformation* of a one-sided error property tester in the random-neighbor model with constant *query* complexity into a one-sided error property tester in the streaming model with constant *space* complexity. Previously such a generic transformation was only known for *bounded-degree* graphs.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Graph property testing, sublinear algorithms, graph streaming algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.16

Category RANDOM

Related Version The full version of the paper is available at `https://arxiv.org/abs/1905.01644`. All the missing proofs can be found in the full version.

Funding *Artur Czumaj*: Research partially supported by the Centre for Discrete Mathematics and its Applications (DIMAP), by IBM Faculty Award, and by EPSRC award EP/N011163/1.
Hendrik Fichtenberger: Research supported by ERC grant No. 307696.
Christian Sohler: Research supported by ERC grant No. 307696.

Acknowledgements We would like to thank anonymous reviewers for extensive comments.

1 Introduction

We consider the fundamental question of understanding the relative power of two important computational models: *property testing* and *data streaming*. We present a novel framework closely linking these areas in the setting of *general graphs* in the context of constant-query complexity testing and constant-space streaming. We first provide a new analysis of constant-query property testers (in the random-neighbor model, see Definition 6) for general graphs


© Artur Czumaj, Hendrik Fichtenberger, Pan Peng, and Christian Sohler;
licensed under Creative Commons License CC-BY

and develop the framework of canonical testers for general graphs. Then, using the concept of canonical testers, we provide a generic *transformation* of a one-sided error property tester in the random-neighbor model with constant *query* complexity into a one-sided error property tester in the streaming model with constant *space* complexity.

Property testing. A fundamental task in the study of big networks/graphs is to efficiently analyze their structural properties. For example, we may want to know if a graph is well-connected, has many natural clusters, has many copies (instances) of some specific sub-structures, etc. Given that modern networks are large, often consisting of millions and billions of nodes (web graph, social networks, etc.), the task of analyzing their structure has become recently more and more challenging, and the running-time efficiency of this task is becoming of critical importance. The framework of *property testing* has been developed to address some of these challenges, aiming to trade the efficiency with the accuracy of the output, with the goal of achieving very fast algorithms.

In (graph) property testing, one of the main challenges is to characterize properties that are testable with a constant number of queries in various computational models. Typically, a tester has query access to a graph (e.g., random vertices or neighbors of a vertex for graphs), and its goal is to determine if the graph satisfies a certain property (e.g., is well-clusterable) or is far from having such a property (e.g., is "far" from any graph being well-clusterable; see, e.g., [18, 19, 20, 39]). To be precise, we define testers as follows. Given a property Π, a tester for Π is a (possibly randomized) algorithm that is given a proximity parameter ε and oracle access to the input graph G. If G satisfies property Π, then the algorithm must accept with probability at least $\frac{2}{3}$. If G is ε-far from Π, then the algorithm must reject with probability at least $\frac{2}{3}$. If the algorithm is allowed to make an error in both cases, we say it is a *two-sided error tester*; if, on the contrary, the algorithm always gives the correct answer when G satisfies the property, we say it is a *one-sided error tester*. Further details of the model depend on the data representation. In the main model considered in this paper, *property testing for general graphs*, we will consider the *random neighbor oracle access* to the input graph (cf. Definition 6), which allows to query a random neighbor of any given vertex[1]. In our model, we will say that G is ε-far from a property Π if any graph that satisfies Π differs from G on at least $\varepsilon|E(G)|$ edges. To analyze the performance of a tester, we will measure its quality in term of its *query complexity*, which is the number of oracle queries it makes.

In the past, a large body of research has focused on the analysis of various graph properties in different graph models, for example, leading to a precise characterization of all properties that can be tested with constant query complexity [1, 3] in the so-called dense model (graphs with $\Theta(n^2)$ edges), and some partial results for bounded-degree graph models (see, e.g., [5, 13, 17, 18, 20, 21, 35]). However, our understanding of the model of general graphs, graphs where each vertex can have arbitrary degree, is still rather limited. We have seen some major advances in testing graph properties for general graphs, including the results of Parnas and

[1] Our model is in contrast with the other two widely used property testing models for graphs with arbitrarily large maximum degree: In the *adjacency list model* [36, 30], the algorithm can perform both *neighbor queries* (i.e., for the i-th neighbor of any vertex v such that $i \leq \deg(v)$), and the *degree queries* (i.e., for the degree $\deg(v)$ of any vertex v); In the *general graph model*, the algorithm is allowed to perform *vertex-pair queries* (i.e., for the existence of an edge between any two vertex pair u, v), in addition to neighbor and degree queries [28, 2, 18]. Still, we believe that the *random neighbor oracle model* considered in this paper is the most natural model of computations in the property testing framework in the context of very fast algorithms, especially those performing $O(1)$ queries. We note however, that our analysis can be generalized to other models of general graphs (cf. the full version).

Ron [36], Kaufman et al. [28], Alon et al. [2], Czumaj et al. [11, 14] (see also the survey in [18, Chapter 10]). The main challenge of the study in the model of general graphs is a lack of good characterization of testable properties and of a good algorithmic toolbox for the problems in this model. Still, the importance of the general graph model and lack of major advances have been widely acknowledged in the property testing community. For example, it is recognized that the general graph model is "most relevant to computer science applications" and "designing testers in this model requires the development of algorithmic techniques that may be applicable also in other areas of algorithmic research" (see [18, Chapter 10.5.3]).

Graph streaming algorithms. One important way of processing large graphs in modern data analysis is to design *graph streaming algorithms* (see, e.g., [31, 34]). A graph streaming algorithm obtains the input graph as a stream of edges in some order and its goal is to process and analyze the input stream in order to compute some basic characteristics about the input graph. For example, we want to know whether the graph is connected, or bipartite, or to know its approximate maximum matching size. Following the mainstream research in data streaming, we focus on algorithms that make only a *single pass* over the graph stream. Since in the single pass model every edge is seen only once, the central complexity measure of data streaming algorithms is the amount of space used to store information about the graph, with the golden standard in streaming being *sublinear space*. Unfortunately, it is known that for many natural graph problems sublinear space $o(n)$ is not possible when the edges are arriving in a single pass and in arbitrary order, where n is the number of vertices of the input graph [23].

There have been several approaches to cope with this inherent limitation of the streaming setting for graph problems. While some of the early works in graph streaming algorithms approached this challenge by allowing more than one pass over the input, the single-pass model is still considered to be the most interesting and the most natural scenario for streaming algorithms. The $\Omega(n)$ space lower bound (e.g., for testing if the graph is connected or estimating the size of transitive closure [23]) led to a significant number of papers designing semi-streaming algorithms, which are algorithms using $O(n \operatorname{polylog}(n))$ space, so only slightly larger than linear in the number of vertices (see the survey [31]). This model leads to sublinear algorithms for dense graphs, where m, the number of edges, is $\omega(n \operatorname{polylog}(n))$. For the very natural setting of *sparse graphs*, semi-streaming algorithms are useless, since with $O(n \operatorname{polylog}(n))$ space one can store the entire input graph (all arriving edges). Therefore, one can trivially solve any graph problem. Some works consider special classes of graphs. For example, it is known how to approximate the matching size within a constant factor in polylogarithmic space for planar graphs or graphs with bounded arboricity (see, e.g., [15, 10, 32, 6]).

Another, central approach to address the linear space lower bounds for graph streaming problems that recently received increasing attention is the *random-order streaming model*, where the edges arrive in random order, i.e., in the order of a uniformly random permutation of the edges (see, e.g., [8, 26, 29, 31, 33, 37, 4, 27, 9, 16]). The assumption about uniformly random or near-uniformly random ordering is very natural and can arise in many contexts. One might also use the random-order streaming model to justify the success of some heuristics in practice, even though there exists strong space lower bound for (the worst case of) the problem. Furthermore, some recent advances have shown that some problems that are hard for adversarial streams can be solved with small space in the random order model. For example, Konrad et al. [29] gave single-pass semi-streaming algorithms for maximum matching for bipartite and general graphs with approximation ratio strictly larger than $\frac{1}{2}$ in the random

order semi-streaming model. Kapralov et al. [26] gave a polylogarithmic approximation algorithm in polylogarithmic space for estimating the size of maximum matching of an unweighted graph in one pass over a random order stream. It is not known if such trade-offs between approximation ratios and space complexity are possible in the adversarial order model. Finally, [37] showed that in the random-order streaming model, even with constant space, one can approximate the number of connected components of the input graph within an additive error of εn, the size of a maximum independent set in planar graphs within a multiplicative factor of $1+\varepsilon$, and the weight of a minimum spanning tree of a connected input graph with small integer edge weights within a multiplicative factor of $1+\varepsilon$. However, for the first and third problems in adversarial order streams, there exist $n^{1-O(\varepsilon)}$ space lower bounds [24]. While these results demonstrate the strength of the random-order streaming model, Chakrabarti et al. [8] proved that $\Omega(n)$ space is needed for any single pass algorithm for graph connectivity in the random-order streaming model, almost matching the optimal $\Omega(n \log n)$ space lower bound in the adversarial order model [40]. This poses a central open question in the area of graph streaming algorithms, of *characterizing graph problems which can be solved with small, sublinear space in the random-order streaming model.*

The main goal of our paper in the context of streaming algorithms, is to address this task and to enlarge the class of graph problems known to be solvable with *small space* in the *random order streaming* model in a *single pass*. Our main focus is on the most challenging scenario: of achieving *constant space*[2].

1.1 Basic Definitions and Overview of Our Results

In this paper, we extend the approach recently introduced by Monemizadeh et al. [33] (see also [37]) to demonstrate a *close connection between streaming algorithms and property testing* in the most general setting of *general graphs*. Monemizadeh et al. [33] show that for *bounded-degree graphs*, any graph property that is constant-query testable in the adjacency list model can be tested with constant space in a single pass in random order streams. In this paper, we show that similar results hold also for general graphs. To this end, we design a novel framework of canonical testers for all constant-query testers for general graphs and apply it to design a generic method of transforming any constant-query tester (with one-sided error) for graph properties into a constant-space tester (with one-sided error) in the random-order streaming model.

We consider the *random neighbor* query oracle model for general graphs, which allows the algorithm to query a random neighbor of any specified vertex (cf. Definition 6).

▶ **Definition 1** (Property testers in the random-neighbor model). *Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be a graph property, where Π_n is a property of graph of n vertices. We say that* Π is testable with query complexity *q, if for every ε and n, there exists an algorithm (called* tester*) that makes at most $q = q(n,\varepsilon)$ oracle queries, and with probability at least $\frac{2}{3}$, accepts any n-vertex graph satisfying* Π*, and rejects any n-vertex graph that is ε-far from satisfying* Π*. If $q = q(\varepsilon)$ is a function independent of n, then we call* Π constant-query testable*. If the tester always accepts graphs that satisfy* Π*, we say that it has* one-sided error*. Otherwise, we say the tester has* two-sided error*.*

[2] Throughout the entire paper, we will count the size of the *space in words* (assuming that a single word can store any single ID of a vertex or of an edge), i.e., space bounds have to be multiplied by $O(\log n)$ to obtain the number of bits used. With this in mind, we use term *constant space* to denote space required to store a constant number of words, or IDs, that is, $O(\log n)$ bits.

We notice that the definition above is generic and can be applied to any of the query oracle models (see e.g. [18]). However, since our main query oracle model is the random-neighbor model, only for that model we will use the terminology from Definition 1 without a direct reference to the query oracle model. We first present *canonical testers* in this model. In order to do so, we introduce a process called *q-random BFS* (q-RBFS) starting with any specified vertex v, i.e., a BFS of depth q that is restricted to visiting at most q random neighbors for every vertex (see Definition 7). We call the subgraph obtained by a q-RBFS a *q-bounded disc*. Our first result is informally stated as follows.

▶ **Theorem 2** (informal; cf. Theorem 10). *If a property $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ is testable with $q = q(\varepsilon)$ queries in the random-neighbor model, then it can also be tested by a canonical tester that*
1. *samples q' vertices;*
2. *performs q'-RBFS from each sampled vertex;*
3. *accepts if and only if the explored subgraph does not contain any (forbidden) graph $F \in \mathcal{F}$,*

where q' depends only on q, and $\mathcal{F}$ is a family of rooted graphs such that each graph $F \in \mathcal{F}$ is the union of q' many q'-bounded discs.

We remark that similar canonical testers have been given for dense graphs [22], bounded-degree graphs and digraphs [21, 12]. Actually, our proof for the above theorem heavily builds upon [21, 12], though our analysis requires some extensions to deal with general graphs (of possibly unbounded degree). To formally state our result regarding testing graph properties in streaming, we introduce the following definition.

▶ **Definition 3** (Property testers in the streaming model). *Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be a graph property, where Π_n is a property of graph of n vertices. We say that Π is testable with space complexity q, if for every ε and n, there exists an algorithm that performs a single pass over an edge stream of an n-vertex graph G, uses $q = q(n, \varepsilon)$* words *of space, and with probability at least $\frac{2}{3}$, accepts G satisfying Π, and rejects G that is ε-far from satisfying Π. If $q = q(\varepsilon)$ is a function independent of n, then we call Π* constant-space testable*. If the tester always accepts the property, then we say that the property can be tested with* one-sided error*. Otherwise, we say the tester has* two-sided error*.*

Our main result and our main technical contribution is the *transformation* of a one-sided error property tester in the random-neighbor model with constant *query* complexity into a one-sided error property tester in the streaming model with constant *space* complexity.

▶ **Theorem 4** (Main Theorem). *Every graph property Π that is constant-query testable with one-sided error in the random-neighbor model is also constant-space testable (space measured in words) with one-sided error in the random order graph streams.*

Applications. We believe that the main contribution of our paper is the general transformation presented in Theorem 4. However, we admit that the number of properties testable with one-sided error with a constant number of queries in the random-neighbor model is rather limited. Still, we can apply our transformation to, for example, the property of being (s, t)-disconnected (i.e., there is no path between s and t), see, e.g., [41][3]. Furthermore, our

[3] The constant-query tester from [41] performs degree queries and neighbor queries, but it is straightforward to simulate it in the random-neighbor model. Indeed, the algorithm in [41] only needs to repeatedly perform a constant-length random walks from s and reject if only if one path from s to t is found. Such an algorithm can be trivially simulated in the random-neighbor model, as each step of a random walk just needs to query one random neighbor of the current vertex.

transformation actually holds when the input graph is restricted to come from a certain class of graphs such as planar graphs, minor-free graphs, or bounded-degree graphs. Since bipartiteness in planar graphs (or minor-free graphs) is testable in the random-neighbor model [11], it is also one-sided error testable in random order streams in constant space; notice that this result stands in contrast to the $n^{1-O(\varepsilon)}$ space lower bound for *adversarial order streams* for (property) testing bipartiteness in planar graphs [24]. Further, recent constant-query complexity testing of H-freeness in planar or minor-free graphs [14] shows that also testing H-freeness is one-sided error testable in random order streams in constant space.

Furthermore, our techniques can also be used to transform any constant-query tester (with one-sided error) in the *random neighbor/edge model* (cf. the full version) to the random-order streaming model, where the random neighbor/edge model allows to sample an edge uniformly at random. Therefore, for example, since the property of being P_k-free (there is no path of length k) is constant-query testable in the random neighbor/edge model with one-sided error [25], P_k-freeness is also constant-space testable with one-sided error in the random order graph streams. Similarly, it is not hard to see that the property of being d-bounded (the maximum degree is at most d) is constant-query testable in the random neighbor/edge model[4], and therefore this property too is constant-space testable with one-sided error in the random order graph streams.

The contribution of our paper goes beyond just establishing a connection between property testing and streaming. While the concept of canonical testers has been used in graph property testing before (cf. [22, 21, 12]), our study and characterization of canonical testers for general graphs (Theorem 2 and Theorem 10) is new. We believe that this study will shed light on our understanding of constant-query testable graph properties and will lead to new results for property testing in general graphs. For example, Czumaj and Sohler [14] recently used our canonical testers as a tool in their proof of a complete characterization of constant-query testable properties in general planar graphs [14] after a preliminary version of this work appeared.

1.2 Challenges and Techniques

The result about constant-space streaming algorithms for bounded-degree graphs by Monemizadeh et al. [33] is obtained by noting that any constant-query complexity tester basically estimates the distribution of local neighborhoods of the vertices (see, e.g., [12, 18, 21]) and emulating any such algorithm on a random order graph stream using constant space. Unfortunately, this approach inherently relies on the assumption that the input graph is of bounded degree. This limitation comes from two ends: on one hand, there has not been known any versatile description of testers for constant-query testable graph properties of general graphs, and on the other hand, the streaming approach from [33] relies on a breadth-first-search-like graph exploration that is possible (with constant space) only when the input graph has no high-degree vertices. A follow-up paper [37] made the first attempt to address the challenge of dealing with general degrees, and considered some problems in which one can *ignore* high degree vertices (e.g., for approximating the number of connected components or the size of a maximum independent set in planar graphs).

[4] If G is ε-far from the property, then at least $\Omega(\varepsilon|E|)$ edges are incident to a node with degree at least $d+1$. Thus, we can simply sample a constant number of edges and check if either of its endpoints has degree at least $d+1$.

One important reason why the earlier approaches have been failing for the model of general graphs, without bounded-degree assumption, was our lack of understanding of constant-query complexity testers in general graphs and the lack of techniques to appropriately emulate off-line algorithms allowing many high-degree vertices. In this paper, we advance our understanding on both of these challenges.

A general and simple canonical tester. To derive a canonical tester for constant-query testable properties in the random-neighbor model, we introduce the process *q-random BFS* (q-RBFS): it starts from any specified vertex v, and then performs a BFS-like exploration of depth q that is restricted to visiting at most q random neighbors at each step (see Definition 7 for the formal definition). We call the subgraph obtained by a q-RBFS a *q-bounded disc*. With the notion of q-RBFS and q-bounded discs, we are able to transform every constant-query tester for properties of general graphs into a *canonical tester* that works as follows: it samples q random vertices, performs a q-RBFS from each sampled vertex, and rejects if and only if the (non-induced) subgraph it has seen (which is a union of q-bounded discs) is isomorphic to some member of a family $\mathcal{F}$ of forbidden subgraphs (see Theorems 2 and 10). Furthermore, such a canonical tester preserves one-sided error, while the query complexity blows up exponentially. We believe that the exponential blow-up is necessary, even for bounded-degree graphs, as adaptivity is essential for property testing in sparse graphs [38, 7]. This is in contrast to the dense graph model for property testing, in which a quadratic blow-up of the query complexity of canonical testers was known [22].

Canonical testers provide us a systematic view of the behavior of constant-query testers in the random-neighbor model. They further tell us that in order to test a constant-query testable property Π, it suffices to estimate the probability that some forbidden subgraph in $\mathcal{F}$ is found by a q-RBFS starting from a randomly sampled vertex. Slightly more formally, we define the *reach probability* of a subgraph $F \in \mathcal{F}$ to be the probability that a q-RBFS starting from a uniformly chosen vertex v sees a graph that is isomorphic to F. In particular, if we can estimate these reach probabilities in random order streams, then we can also test Π accordingly.

The problem with this approach is that it is hard to estimate the reach probabilities of subgraphs in $\mathcal{F}$. The main challenge here is that a forbidden subgraph $F \in \mathcal{F}_n$ may be the union of more than two or more subgraphs obtained from different q-RBFS that may intersect with each other.

A refined canonical tester. To cope with the challenge mentioned above of estimating the reach probabilities of subgraphs in $\mathcal{F}$, we decompose each forbidden subgraph $F \in \mathcal{F}_n$ into all possible sets of intersecting q-bounded discs whose union is F and then try to recover F from these sets. In order to recover F from such a decomposition, we have to identify and monitor *vertices that are contained in more than one q-bounded disc of F*.

We refine the analysis of the canonical tester and separate the q-bounded discs explored by each q-RBFS and keep track of their intersections (cf. Theorem 17). We first observe that for every input graph G and every ε, there exists a *small fixed set* $V_\alpha \subseteq V$ of all vertices whose probability to be visited by a random q-RBFS from a random vertex exceeds some small threshold α (depending on q and ε, but independent of n). In other words, with constant probability, the subgraphs explored by multiple q-RBFS in the canonical tester will only overlap on vertices from V_α. Furthermore, we prove that the degree of all vertices in V_α is at least linear (in n), and with constant probability, two random q-RBFS subgraphs will not share any edge. Since V_α has constant size, each q-bounded disc can be viewed as a

colored q-bounded disc *type* such that each vertex in V_α is assigned a unique color from a constant-size palette. This way, it is possible to reversibly decompose each $F \in \mathcal{F}_n$ into a multiset of colored q-bounded disc types (actually, there may be many such multisets for each F): since the q-bounded discs that are explored by different q-RBFS intersect only at vertices in V_α, F is obtained by identifying vertices of the same color. See Figure 1 in Appendix A for an example.

These properties are crucial to describe the forbidden subgraphs in terms of the graphs seen by the q many q-RBFS that the canonical tester performs and a *constant-size* description of their interaction, i.e., how they overlap.

Simulation in the streaming. In the streaming, in order to simulate q-RBFS, it is natural to consider the following procedure called StreamCollect (q-SC, see Algorithm 2 in Appendix B) to explore the subgraph surrounding any specified vertex. That is, it maintains a connected component C that initially contains only the start vertex. Whenever it reads an edge that connects to the current C and the augmented component may be observed by a run of q-RBFS, it adds the edge to C.

Note that one important feature of random order streams is that we would see the right exploration (as in the query model) with constant probability, while it is challenging to verify if the subgraph we collected from the stream is indeed the right exploration (cf. [33, 37] for a more detailed discussion). In our setting, as we mentioned, another technical difficulty is to analyze whether subgraphs found by running the stream procedure multiple times *intersect* in exactly the same way as the q-bounded discs that are found by q-RBFS.

With the refined canonical tester, which specifies how different q-RBFS procedures intersect, we are able to *simulate one-sided error constant-query testers* in the random-neighbor model for general graphs in the *random-order streaming model*. Since the considered property Π is one-sided error testable in the random-neighbor model, it suffices to detect a forbidden subgraph F in the family $\mathcal{F}$ corresponding to Π with constant probability. That is, it suffices to show that if the graph is far from having the property, then for any forbidden subgraph H that can be reached by the canonical tester with probability p, it can also be detected by *multiple* StreamCollect subroutines with probability at least cp for some suitable constant c.[5]

In order to do so, we first decompose the forbidden subgraphs that characterize the property into colored subgraphs, where each subgraph corresponds to a run of q-RBFS and vertices in V_α are colored with a unique color. Then, we prove that for a sufficiently large sample of vertices, the q-SC subroutines starting from these sampled vertices will collect, for each colored subgraph H, at least as many instances of H as the canonical property tester sees. Suppose that the input graph is far from the property. Since the subgraphs observed by the canonical tester intersect only at vertices in V_α, i.e., colored vertices, with constant probability, it is possible to stitch a forbidden subgraph by identifying vertices of the same color in the analysis.

[5] Note that this is not sufficient for simulating two-sided error testers. Let us take the property connectivity (which is 2-sided error testable in random-neighbor model) for example. If the input graph is a path on n vertices, then a q-RBFS will detect a forbidden subgraph (i.e., a path of constant length that is not connected to the rest) corresponding to connectivity with small constant probability, while a q-SC might see a forbidden subgraph with high constant probability. That is, in order to test connectivity, we need to be able to approximate the *frequencies* of the forbidden subgraphs, for which our current techniques fail.

The analysis of this procedure is two-fold. First, we show that if a single run of q-RBFS from v sees a certain colored q-bounded disc type with probability p (where the colored vertices are V_α), then a single run of q-SC from v sees this disc type with probability cp for some suitable constant c (see Corollary 20).

The second step (which is the main technical part) is to show that if the probability that a q-RBFS from a random vertex sees a colored q-bounded disc type Δ is p, then with constant probability, for a sufficiently large sample set S, the calls to q-SC from vertices in S will also see a q-bounded disc type Δ, even though there are intersections from different q-SCs (see Lemma 21). Then we can show that if the input graph is far from the property, with constant probability, we can stitch the colored q-bounded discs to obtain a forbidden subgraph $F \in \mathcal{F}$ (see Theorem 4).

Finally, we remark that colors are only used in the analysis as the streaming algorithm can identify intersections of multiple q-SC by the vertex labels. However, the colors are crucial to the analysis: without colors, we cannot guarantee that the q-bounded disc types found by multiple q-SCs can be stitched in the same way as the q-bounded disc types found by q-RBFS. Here is an example: Consider some constant-query testable property Π such that the set of forbidden subgraphs $\mathcal{F}$ contains a graph F that is not a subgraph of any single q-bounded disc type (i.e, it is the union of at least two intersecting q-bounded disc types). For the sake of illustration, a concrete example is provided in Figure 2 in Appendix A. In order to reject, the canonical property tester needs to find at least two intersecting q-bounded discs such that their union contains F as a subgraph. However, even if we bound, for each *uncolored* q-bounded disc type Δ, the probability that q-SC finds Δ by some constant fraction of the probability that q-RBFS finds Δ, this is not sufficient to conclude that the probability that multiple q-SCs find a copy of F is bounded by a constant fraction of the probability that multiple q-RBFS find a copy of F. The reason is that q-SC might only find copies of Δ that are not intersecting, while q-RBFS might tend to find copies of Δ that intersect. Again, see Figure 2 for an example. Therefore, we need to preserve, for each q-bounded disc type Δ, the information which of the corresponding vertices in the input graph are likely to be contained in more than one q-RBFS for the analysis.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph. We will assume that the vertex set V of G is $[n] = \{1, \ldots, n\}$, and we let $\deg(v)$ denote the degree of $v \in V$. Sometimes, we use $V(G)$ to denote the vertex set V of G and $E(G)$ to denote the edge set E of G. We let $\mathcal{S}(G)$ denote the input stream of edges that defines G. In this paper, we consider streaming algorithms for random order streams, i.e., the input stream $\mathcal{S}(G)$ to our algorithm is drawn uniformly from the set of all permutations of E. We are interested in *streaming algorithms that have constant space complexity* in the size of the graph, where we count the size of the space in words, i.e., space bounds have to be multiplied by $O(\log n)$ to obtain the number of bits used, see also Footnote 2.

A graph G is called a *rooted* graph if at least one vertex in G is marked as *root*. Let us define the notion of a root-preserving isomorphism.

▶ **Definition 5.** *Given two rooted graphs H_1 and H_2, a* root-preserving isomorphism *from H_1 to H_2 is a bijection $f : V(H_1) \to V(H_2)$ such that 1) if u is the root of $V(H_1)$ then $f(u)$ is the root of $V(H_2)$, and 2) that $(u, v) \in E(H_1)$ if and only if $(f(u), f(v)) \in E(H_2)$. If there is a root-preserving isomorphism from H_1 to H_2 then we say that H_1 is root-preserving isomorphic to H_2 and denote it by $H_1 \simeq H_2$.*

3 Canonical Constant-Query Testers in General Graphs

In this section, we present our main result on the *canonical testers for constant-query testable properties in general graphs*. After starting with some basic definitions, we will present two canonical testers for constant-query testable properties in general graphs. Our first canonical tester is of a general form (see Section 3.2) and our second tester (see Theorem 17 in Section 3.3) is slightly more refined, allowing for a more natural use later in the setting of streaming algorithms in Section 5.

We note that in this paper we focus on one specific model of access to the input graph, the *random-neighbor model*. It is possible to extend some of our analysis (of canonical testers) to some other graph access models, though (cf. the full version).

3.1 Random BFS and Bounded Discs

Property testing in query oracle model. Since we consider general graphs, without any bounds for vertex degrees, we have to carefully define the access provided to the input graph in the property testing framework. The access to the input graph is given by *queries* to an *oracle* representing the graph. There have been several oracles considered in the literature for general graphs, but our main focus is on the *random-neighbor model*, which we consider to be natural for graphs with unbounded degree, especially in the context of properties testable with a constant number of queries.

▶ **Definition 6** (Random-neighbor model)**.** *In the* random-neighbor model*, an algorithm is given $n \in \mathbb{N}$ and access to an input graph $G = (V, E)$ by a query oracle, where $V = [n]$. The algorithm may ask queries based on the entire knowledge it has gained by the answers to previous queries. The* random neighbor query *specifies a vertex $v \in V$ and the oracle returns a vertex that is chosen i.u.r. (independently and uniformly at random) from the set of all neighbors of v.*

Notice that in the random-neighbor model, since $V = [n]$, the algorithm can also trivially select a vertex from V i.u.r. We believe that the random-neighbor model is the most natural model of computations in the property testing framework in the context of very fast algorithms (especially those of constant query complexity), and therefore our main focus is on that model. However, we want to point out that some of our results are sufficiently general to apply to a larger variety of the query oracle models, though we will not elaborate about it here (cf. the full version).

We describe the first canonical testers of all constant-query testers (in the random-neighbor model) for general graphs, both, for one-sided and two-sided errors. With this canonization, we can model all graph properties testable with a constant number of queries using *canonical testers*; see Theorems 10 and 17 for formal statements.

To formalize our canonical testers for all constant-query testers in the random-neighbor model, we will use the following two definitions of constrained random BFS-like graph exploration and of bounded discs.

We begin with the definition of a q-RBFS process, which starts at some vertex and explores its neighborhood in a BFS-like fashion, conditioned on a bound of the depth and the breadth of the exploration (see Definition 7 for formal definition and Algorithm 1 in Appendix B for the detailed implementation).

▶ **Definition 7** (q-random BFS)**.** *Let $q > 0$ be an integer and G be a simple graph. For any vertex $v \in V(G)$, the* q-random BFS *(abbreviated as* q-RBFS*) explores a random subset of the q-neighborhood of v in G iteratively as follows. First, it initializes a queue $Q = \{v\}$*

and a graph $H = (\{v\}, \emptyset)$. Then, in every iteration, it pops a vertex u from Q and samples q random neighbors $s_{u,1}, \ldots, s_{u,q}$ of u. For every edge $e = \{u, s_{u,i}\}$, it adds $s_{u,i}$ and the directed edge $(u, s_{u,i})$ to H. Furthermore, if $s_{u,i}$ has distance less than q from v in H and $s_{u,i}$ has not been added to Q before, $s_{u,i}$ is appended to Q. When Q is empty, all edges in H are made undirected (without creating parallel edges) and H is returned.

Any output of q-RBFS algorithms can be described in a static form using the concept of bounded discs.

▶ **Definition 8** (q-bounded disc). *For a given $q \in \mathbb{N}$, graph $G = (V, E)$, and vertex $v \in V$, a* q-bounded disc of v in G *is any subgraph H of G that is rooted at v and can be returned by $\textsc{RandomBFS}(G, v, q)$. In this case, vertex v is called a* root *of the q-bounded disc H and the maximum distance from v to any other vertex in H is called the* radius *of H.*

All q-bounded discs that are root-preserving isomorphic form an equivalence class.

▶ **Definition 9** (q-bounded disc type). *Let H be a q-bounded disc. The equivalence class of H with respect to $\simeq$, i. e., the existence of a root-preserving isomorphism (see Definition 5), is called the q-bounded disc type of H.*

3.2 Canonical Testers: A General Version

Now we present the proof of our first main result. We show that any tester with query complexity $q = q(\varepsilon, n)$ in the random-neighbor model can be simulated by a *canonical tester* that samples $q' = O(q)$ vertices and rejects if and only if the union of the subgraphs induced by the q'-RBFS from the sampled vertices belongs to some family of forbidden graphs.

▶ **Theorem 10** (Canonical tester). *Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be a graph property that can be tested in the random-neighbor model with query complexity $q = q(\varepsilon, n)$ and error probability at most $\frac{1}{3}$. Then for every ε, there exists an infinite sequence $\mathcal{F} = (\mathcal{F}_n)_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$,*

- *$\mathcal{F}_n$ is a set of rooted graphs such that each graph $F \in \mathcal{F}_n$ is the union of q' many q'-bounded discs;*
- *the property Π_n on n-vertex graphs can be tested with error probability at most $\frac{1}{3}$ by the following canonical tester:*
  1. *sample q' vertices i.u.r. and mark them roots;*
  2. *for each sampled vertex v, perform a q'-RBFS starting at v;*
  3. *reject if and only if the explored subgraph is root-preserving isomorphic to some $F \in \mathcal{F}_n$,*

where $q' = cq$ for some constant $c > 1$. The query complexity of the canonical tester is $q^{O(q)}$. Furthermore, if $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ can be tested in the random-neighbor model with one-sided error, then the resulting canonical tester for Π has one-sided error too, i.e., the tester always accepts graphs satisfying Π.

3.3 Canonical Testers Revisited: Identifying Vertices in the Intersecting Discs

Theorem 10 provides us a canonical way of testing constant-query testable properties (in the random-neighbor model) by relating the tester to a set of forbidden subgraphs $\mathcal{F}_n$ for every $n \in \mathbb{N}$. However, as we mentioned in Section 1, it is hard to directly use Theorem 10 to design and analyze our streaming testers due to the intersections of q-RBFS. In order to tackle this difficulty, we decompose each forbidden subgraph $F \in \mathcal{F}_n$ into all possible sets of intersecting q-bounded discs whose union is F. In order to recover F from such a decomposition, we have to identify and monitor *vertices that are contained in more than one q-bounded disc of F.*

Identifying vertices with large reach probability. Now we prove that with constant probability the q-bounded discs found by q-RBFS will only intersect on a *small* set of vertices V_α and the discs will not intersect on any edge.

We begin with a useful definition on the probability of reaching a vertex from a q-RBFS.

▶ **Definition 11.** *For each vertex v, the* reach probability $r(v) := r_q(v)$ *of v is the probability that a q-RBFS starting at a uniformly randomly chosen vertex reaches v.*

In the following lemma, we give an upper bound on the size of the set of vertices with constant reach probability, which also implies that with constant probability, the number of vertices visited by at least two q-RBFS that the canonical tester performs is small. For any α, $0 \leq \alpha \leq 1$, we let $V_\alpha := \{v \in V : r(v) \geq \alpha\}$. For a fixed q, let $c_j := \sum_{i=0}^{j} q^i = \frac{q^{j+1}-1}{q-1}$.

▶ **Lemma 12.** *For any $0 < \alpha < 1$, it holds that $|V_\alpha| \leq \frac{c_q}{\alpha}$.*

We further show that with high probability, two q-RBFS starting from vertices chosen i.u.r. will not share an edge (i.e., will not visit the same edge).

▶ **Lemma 13.** *Let $0 < \alpha \leq 1$. Let $n \geq \frac{qc_q}{\alpha^2}$. Let u, v be two randomly chosen vertices. Let H_u and H_v denote the subgraphs visited by two q-RBFS starting at u and v, respectively. Then with probability at least $1 - qc_q \cdot 2\alpha$, no edge will be contained in both H_u and H_v.*

Colored q-bounded disc types. To identify vertices in V_α, we assign them unique colors for the analysis. We call a disc *r-colored* if in addition to uncolored vertices in the disc, some vertices in the disc may be colored with at most r colors, each color being used at most once. Two colored q-bounded disc types Δ_1 and Δ_2 (cf. Definition 9) are called to be isomorphic to each other, denoted by $\Delta_1 \simeq \Delta_2$, if there is a root-preserving isomorphism f from Δ_1 to Δ_2 that also preserves the colors, i.e., if and only if $u \in V(\Delta_1)$ is colored with color c, then $f(u) \in \Delta_2$ is colored with color c.

▶ **Definition 14.** *Let $q > 0$ be an integer. We let $\mathcal{H}_q := \{\Delta_1, \cdots, \Delta_N\}$ denote the set of all possible r-colored q-bounded disc types, where N is the total number of such types.*

For any given colored q-bounded disc type, we have the following definition on the probability of seeing such a disc type from a q-RBFS.

▶ **Definition 15** (Reach probability of colored q-bounded disc types)**.** *Let $G = (V, E)$ be a graph with n vertices such that each vertex in V_α is assigned to a unique color. Let $\Delta \in \mathcal{H}_q$ be a colored q-bounded disc type. The* reach probability of Δ in G *is the probability that a q-RBFS from a random vertex in G reveals a graph that is (root- and color-preserving) isomorphic to Δ, that is* $\mathrm{Reach}_G(\Delta) := \Pr_{v \sim V, BFS}[\textsc{RandomBFS}(G, v, q) \simeq \Delta]$.

For a given vertex v, the reach probability of Δ from v in G is the probability that a q-RBFS from v in G induces a graph that is (root- and color-preserving) isomorphic to Δ, that is $\mathrm{Reach}_G(v, \Delta) := \Pr_{BFS}[\textsc{RandomBFS}(G, v, q) \simeq \Delta]$.

Recall from Definition 8 that a q-bounded disc of v in G is any subgraph H of G that is rooted at v and can be returned by $\textsc{RandomBFS}(G, v, q)$. In order to estimate the reach probability of a colored q-bounded disc type, we consider for each starting vertex v, the set of all possible colored q-bounded discs, denoted $\mathcal{C}_v$, that one can see from a q-RBFS from v.

▶ **Definition 16** (Reach probability of a colored q-bounded disc)**.** *Let $G = (V, E)$ be a graph in which all vertices in V_α are uniquely colored. Let v be a vertex in G. A* colored q-bounded disc *of v is a q-bounded disc of v in G in which all vertices in V_α colored. We let $\mathcal{C}_v$ denote*

the set of all possible colored q-bounded discs of v.[6] *For any fixed colored q-bounded disc $C \in \mathcal{C}_v$ of v, the reach probability of C from v is the probability that a q-RBFS from v sees exactly C, that is,* $\mathrm{Reach}_G(v, C) := \Pr_{BFS}[\textsc{RandomBFS}(G, v, q) = C]$.

By our definition, the q-RBFS from a vertex v in the colored graph G (with vertices in V_α colored) will return exactly one colored q-bounded disc of v. For each colored q-bounded disc type Δ, we let $\mathcal{C}_v(\Delta)$ denote the subset of $\mathcal{C}_v$ which contains all colored q-bounded discs of v that are isomorphic to Δ. Therefore, we have the following observation: $\mathrm{Reach}_G(v, \Delta) = \sum_{D \in \mathcal{C}_v(\Delta)} \mathrm{Reach}_G(v, D)$.

Canonical testers with distinguished vertices in the intersecting discs. Now, we give a refined characterization of the family of forbidden subgraphs corresponding to any constant-query testable property in general graphs, which establishes the basis of our framework for transforming the canonical constant-query testers in the random-neighbor model to the random-order streaming model.

In our next theorem, we will consider partially vertex-colored graphs and q-bounded discs: we color each vertex in V_α with a unique color from a palette of size $|V_\alpha|$. Recall from Lemma 12 that $|V_\alpha| \leq \frac{c_q}{\alpha}$. We obtain canonical testers of constant-query testable properties by *forbidden colored q-bounded discs* instead of *forbidden subgraphs* (that can be composed of more than a single q-bounded disc). See Figure 1 in Appendix A for an example.

▶ **Theorem 17.** *Let $\Pi = (\Pi_n)_{n\in\mathbb{N}}$ be a graph property that is testable with query complexity $q = q(\varepsilon)$. Let $\alpha \leq \frac{1}{24q'c_{q'}}$, where q' is the number from Theorem 10 for a canonical tester with error probability $1/6$. There is an infinite sequence $\mathcal{F}' = (\mathcal{F}'_n)_{n\in\mathbb{N}}$ such that for any $\epsilon > 0$, $n \geq \frac{q'c_{q'}}{\alpha^2}$, the following properties hold:*

- *$\mathcal{F}'_n$ is a set of graphs, and for each graph $F \in \mathcal{F}'_n$, there exists at least one multiset S of q' many $c_{q'}/\alpha$-colored and rooted q'-bounded disc types such that 1) the disc types are pairwise edge-disjoint, and 2) the graph obtained by identifying all vertices of the same color in the bounded discs of S is isomorphic to F.*
- *For any n-vertex graph $G = (V, E)$ such that each vertex in V_α is colored uniquely, let $S_{q'}$ denote the set of q' subgraphs obtained by performing q'-RBFS starting at q' vertices sampled i.u.r. Then,*
  - *if $G \in \Pi_n$, with probability at least $\frac{2}{3}$, there is no $F \in \mathcal{F}'_n$ such that F is isomorphic to a graph from $S_{q'}$,*
  - *if G is ε-far from Π_n, with probability at least $\frac{2}{3}$, there exists $F \in \mathcal{F}'_n$ such that F is isomorphic to a graph from $\simeq S_{q'}$,*

  where the probability is taken over the randomness of $S_{q'}$.

Furthermore, if Π can be tested with one-sided error, then for $G \in \Pi_n$, with probability 1, there is no $F \in \mathcal{F}'_n$ such that $F \simeq S_{q'}$.

4 Estimating the Reach Probabilities in Random Order Streams

Given a canonical tester $\mathcal{T}$ for a property Π that is constant-query testable in the random-neighbor model, we transform it into a random-order streaming algorithm as follows. Recall from Theorem 10 that $\mathcal{T}$ explores the input graph by sampling vertices uniformly at random and running q-RBFS for each of these vertices. Only if the resulting subgraph contains an

[6] Note that the number $|\mathcal{C}_v|$ of colored q-bounded discs of v can be a polynomial of n.

instance of a forbidden subgraph from a family $\mathcal{F}$, it rejects. It seems natural to define a procedure like q-RBFS for random order streams, namely a procedure StreamCollect($\mathcal{S}(G), v, q$) (q-*SC*), and let the streaming algorithm reject only if the union of all q-SC contains an instance of a graph from $\mathcal{F}$. However, this raises a couple of issues.

It seems hard to analyze the union of the subgraphs obtained by q-SC and relate it to the union of subgraphs observed by q-RBFS because the interference between two q-SC is quite different from the interference of two q-RBFS. Therefore, we use Theorem 17, which roughly says that we can decompose each forbidden subgraph into colored q-bounded disc types. This leads to the following idea: First, we prove that for any colored q-bounded disc type Δ, if q-RBFS finds an instance of Δ in the input graph with probability p (where colors correspond to intersections of multiple RBFS), then q-SC finds an instance of Δ with probability cp for some suitable constant c. Then, we prove that if S is a sufficiently large set of vertices sampled uniformly at random, for each colored q-bounded disc type Δ, the fraction of q-bounded discs found by q-SCs started from S that are isomorphic to Δ is bounded from below by the probability that a q-RBFS from a random vertex sees a colored q-bounded disc that is isomorphic to Δ. Finally, in the next section, we conclude that if q-RBFS finds a forbidden subgraph $F \in \mathcal{F}$ with probability p, then the fraction of q-SC also finds this subgraph with probability cp (for some suitable constant c) because it will find the corresponding colored q-bounded discs that assemble F.

Collecting a q-Bounded Disc in a Stream. In our streaming algorithm, we need to collect a q-bounded disc from a vertex v. We do this in a natural and greedy way: We start with a graph $H = (U, F)$ with $U = \{v\}$ and $F = \emptyset$. Then whenever we see an edge (u, w) from the stream that is connected to our current graph H and adding (u, w) to H does not violate the q-bounded radius of H, and the degree of u or the degree of w in H is still less than q^{2q}, we add it to F (and possibly add one of its endpoints to U); otherwise, we simply ignore the edge. Note that the algorithm does not assign colors to the subgraphs it explores. The procedure StreamCollect is formally defined in Algorithm 2 in Appendix B.

Relation of One q-SC and One q-RBFS. In the following, we show that for any vertex v, and any colored q-bounded disc C of v, the probability of collecting C from v by running StreamCollect on a random order edge stream is at least a constant factor of the probability of reaching C from v by running a q-RBFS on G. The statements in this subsection hold for a single run of q-SC.

We emphasize that the coloring does not need to be explicitly given. It is sufficient if it can be applied when random access to the graph is given. In particular, we may assign each vertex in V_α a unique color. This enables us to identify the vertices where multiple q-RBFS may intersect, which is crucial to apply Theorem 17 later.

▶ **Lemma 18.** *Let G be a vertex-colored graph. There exists a constant $c_*(q)$ depending on q, such that for any colored q-bounded disc C of G, it holds that the probability (over $\mathcal{S}(G)$) that StreamCollect($\mathcal{S}(G), v, q$) contains C is at least $c_*(q) \cdot \mathrm{Reach}_G(v, C)$.*

The following lemma performs the step from q-bounded discs to q-bounded disc *types*.

▶ **Lemma 19.** *Let Δ be a fixed colored q-bounded disc type. Let $X_{v,\Delta}$ denote the indicator variable that StreamCollect from v collects a subgraph that contains a colored q-bounded disc of v that is isomorphic to Δ. Let Y_v denote the indicator variable that RandomBFS from v sees a colored q-bounded disc of v that is isomorphic to Δ. Then it holds that $\mathrm{E}_{\mathcal{S}(G)}[X_{v,\Delta}] \geq c_*(q) \cdot \mathrm{E}_{RBFS}[Y_v]$, where $c_*(q)$ is the constant from Lemma 18.*

Now we consider the probability of seeing a colored q-disc type Δ. Note that $\mathrm{E}_{\mathcal{S}(G)}[X_{v,\Delta}] = \Pr_{\mathcal{S}(G)}[\text{StreamCollect}(\mathcal{S}(G), v, q)$ contains a subgraph F with $F \simeq \Delta]$. Furthermore, it holds that $\mathrm{E}_{RBFS}[Y_v] = \mathrm{Reach}_G(v, \Delta)$. Thus, we have the following lemma.

▶ **Corollary 20.** *For any colored q-bounded disc type Δ, the probability (over $\mathcal{S}(G)$) that* StreamCollect*$(\mathcal{S}(G),v,q)$ contains a subgraph F with $F \simeq \Delta$ is at least $c_*(q) \cdot \mathrm{Reach}_G(v, \Delta)$.*

Relation of Multiple q-SCs and q-RBFS. In the above, we related a single run of q-RBFS and a single run of q-SC. In particular, Corollary 20 states that if a q-RBFS starting from v finds some colored q-bounded disc type Δ with probability p, q-SC finds the same type Δ with probability $\Omega(p)$. However, the forbidden subgraphs that the property tester aims to find may be composed of more than one q-bounded disc. Therefore, we need to prove that if multiple runs of q-RBFS find q-bounded disc types $\Delta_1, \ldots, \Delta_k$ whose union contains an instance of a forbidden subgraph $F \in \mathcal{F}'_n$, then multiple runs of q-SC will find $\Delta_1, \ldots, \Delta_k$ with probability $\Omega(p)$.

We now show our main technical lemma on estimating the reach probability of q-bounded disc types in random order streams. Again, the coloring of vertices in G is implicit and only used for the analysis.

▶ **Lemma 21.** *Let $G = (V, E)$ be a graph defined by a random order stream and let all vertices in V_α be colored. Let $q > 0$ be an integer and let $c'_q := \sum_{i=0}^{q+1} q^{2qi}$. Let $\delta > 0$, and let S denote a set of vertices that are chosen uniformly, where $s := |S| \geq \max\left\{\frac{1}{20\sqrt{\alpha q^{2q} \cdot c'_q}}, \frac{5000|\mathcal{H}_q|}{c_*(q)\delta^3}\right\}$, $\alpha := \frac{c_*(q)^4\delta^8}{10^9|\mathcal{H}_q|^2 q^{2q} c'_q}$. Let $\mathcal{J} := \{H_v : H_v =$ StreamCollect$(\mathcal{S}(G), v, q), v \in S\}$ denote the set of colored q-bounded discs collected by* StreamCollect *from vertices in S. For each type $\Delta \in \mathcal{H}_q$, let X_Δ denote the number of graphs H in $\mathcal{J}$ such that H contains a subgraph F with $F \simeq \Delta$.*

Then it holds that with probability at least $1 - \frac{1}{100}$, for each type $\Delta \in \mathcal{H}_q$, $q_\Delta := \frac{1}{c_*(q)} \cdot \frac{X_\Delta}{s} \geq \mathrm{Reach}_G(\Delta) - \delta$, where $c_*(q)$ is a constant from Corollary 20.

5 Testing Graph Properties in Random Order Streams

Now we transform constant-query property testers (with one-sided error) into constant-space streaming property testers, and prove Theorem 4. The main idea is to explore the streamed graph by StreamCollect and look for the forbidden subgraphs in $\mathcal{F}_n$ that characterize Π (see Theorem 10). However, in the underlying analysis, we use the (reversible) decomposition of the forbidden subgraphs in $\mathcal{F}_n$ into $\mathcal{F}'_n$ (see Theorem 17) to prove the following: if $\mathcal{T}$ finds the colored q-bounded discs $\Delta_1, \ldots, \Delta_k$ that compose a forbidden subgraph $F \in \mathcal{F}'_n$ with probability p, then the streaming tester will find at least as many copies of $\Delta_1, \ldots, \Delta_k$ as $\mathcal{T}$ (see Lemma 21) and can stitch F from these copies. With these tools at hand, we can incorporate our analysis from previous sections to complete the proof of Theorem 4 (see Appendix C).

6 Conclusions

We gave the first canonical testers for all constant-query testers in the random-neighbor model for general graphs and show that one can emulate any constant-query tester with one-sided error in this query model in the random-order streaming model with constant space. Our transformation between constant-query testers and streaming algorithms with constant space provides a strong and formal evidence that property testing and streaming algorithms are very closely related. Our results also work for any restricted class of general graphs and other query models, e.g., random neighbor/edge model. It follows that many properties are constant-space testable (with one-sided error) in random order streams, including (s,t)-disconnectivity, being d-bounded degree, k-path-freeness of general graphs and bipartiteness and H-freeness of planar (or minor-free) graphs.

References

1 Noga Alon, Eldar Fischer, Ilan Newman, and Asaf Shapira. A combinatorial characterization of the testable graph properties: It's all about regularity. *SIAM Journal on Computing*, 39(1):143–167, 2009. doi:10.1137/060667177.

2 Noga Alon, Tali Kaufman, Michael Krivelevich, and Dana Ron. Testing triangle-freeness in general graphs. *SIAM Journal on Discrete Mathematics*, 22(2):786–819, 2008. doi:10.1137/07067917X.

3 Noga Alon and Asaf Shapira. A characterization of the (natural) graph properties testable with one-sided error. *SIAM Journal on Computing*, 37(6):1703–1727, 2008. doi:10.1137/06064888X.

4 Sepehr Assadi, MohammadHossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein. Coresets meet EDCS: Algorithms for matching and vertex cover on massive graphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1616–1635, 2019.

5 Itai Benjamini, Oded Schramm, and Asaf Shapira. Every minor-closed property of sparse graphs is testable. *Advances in Mathematics*, 223(6):2200–2218, 2010.

6 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.

7 Clément L. Canonne and Tom Gur. An adaptivity hierarchy theorem for property testing. *Computational Complexity*, 27(4):671–716, 2018. doi:10.1007/s00037-018-0168-4.

8 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 (STOC)*, pages 641–650, 2008.

9 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1786–1802, 2020.

10 Graham Cormode, Hossein Jowhari, Morteza Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *Proceedings of 25th Annual European Symposium on Algorithms (ESA)*, pages 29:1–29:15, 2017. doi:10.4230/LIPIcs.ESA.2017.29.

11 Artur Czumaj, Morteza Monemizadeh, Krzysztof Onak, and Christian Sohler. Planar graphs: Random walks and bipartiteness testing. In *Proceedings of the 52nd Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 423–432, 2011. doi:10.1109/FOCS.2011.69.

12 Artur Czumaj, Pan Peng, and Christian Sohler. Relating two property testing models for bounded degree directed graphs. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1033–1045, 2016.

13 Artur Czumaj, Asaf Shapira, and Christian Sohler. Testing hereditary properties of nonexpanding bounded-degree graphs. *SIAM Journal on Computing*, 38(6):2499–2510, 2009.

14 Artur Czumaj and Christian Sohler. A characterization of graph properties testable for general planar graphs with one-sided error (It's all about forbidden subgraphs). In *Proceedings of the 60th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 1513–1536, 2019.

15 Hossein Esfandiari, Mohammadtaghi Hajiaghayi, Vahid Liaghat, Morteza Monemizadeh, and Krzysztof Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. *ACM Transactions on Algorithms*, 14(4):48, 2018.

16 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1773–1785, 2020.

17 Hendrik Fichtenberger, Pan Peng, and Christian Sohler. Every testable (infinite) property of bounded-degree graphs contains an infinite hyperfinite subproperty. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, Proceedings, pages 714–726, 2019. doi:10.1137/1.9781611975482.45.

18 Oded Goldreich. *Introduction to Property Testing*. Cambridge University Press, 2017.

19 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.

20 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32:302–343, 2002.

21 Oded Goldreich and Dana Ron. On proximity-oblivious testing. *SIAM Journal on Computing*, 40(2):534–566, 2011.

22 Oded Goldreich and Luca Trevisan. Three theorems regarding testing graph properties. *Random Structures & Algorithms*, 23(1):23–57, 2003.

23 Monika Rauch Henzinger, Prabhakar Raghavan, and Sridhar Rajagopalan. Computing on data streams. In *Proceedings of the DIMACS Workshop on External Memory Algorithms*, pages 107–118, 1998.

24 Zengfeng Huang and Pan Peng. Dynamic graph stream algorithms in $o(n)$ space. *Algorithmica*, 81(5):1965–1987, 2019.

25 Kazuo Iwama and Yuichi Yoshida. Parameterized testability. *ACM Transactions on Computation Theory*, 9(4):16, 2018.

26 Michael Kapralov, Sanjeev Khanna, and Madhu Sudan. Approximating matching size from random streams. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 734–751, 2014.

27 Michael Kapralov, Slobodan Mitrović, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1753–1772, 2020.

28 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/S0097539703436424.

29 Christian Konrad, Frédéric Magniez, and Claire Mathieu. Maximum matching in semi-streaming with few passes. In *Proceedings of the 15th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 231–242, 2012.

30 Mitsuru Kusumoto and Yuichi Yoshida. Testing forest-isomorphism in the adjacency list model. In *Proceedings of the 37th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 763–774, 2014.

31 Andrew McGregor. Graph stream algorithms: A survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

32 Andrew McGregor and Sofya Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *Proceedings of the 1st Symposium on Simplicity in Algorithms (SOSA)*, pages 14:1–14:4, 2018. doi:10.4230/OASIcs.SOSA.2018.14.

33 Morteza Monemizadeh, S. Muthukrishnan, Pan Peng, and Christian Sohler. Testable bounded degree graph properties are random order streamable. In *Proceedings of the 44th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 131:1–131:14, 2017.

34 Shanmugavelayutham Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005.

35 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013.

36 Michal Parnas and Dana Ron. Testing the diameter of graphs. *Random Structures & Algorithms*, 20(2):165–183, 2002. doi:10.1002/rsa.10013.

37 Pan Peng and Christian Sohler. Estimating graph parameters from random order streams. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2449–2466, 2018.

38 Sofya Raskhodnikova and Adam Smith. A note on adaptivity in testing properties of bounded degree graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 13(089), 2006.

39 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

40 Xiaoming Sun and David P Woodruff. Tight bounds for graph problems in insertion streams. In *Proceedings of the 18th International Conference on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, pages 435–448, 2015.

41 Yuichi Yoshida and Yusuke Kobayashi. Testing the (s,t)-disconnectivity of graphs and digraphs. *Theoretical Computer Science*, 434:98–113, 2012.

A Missing Illustrations from Section 1

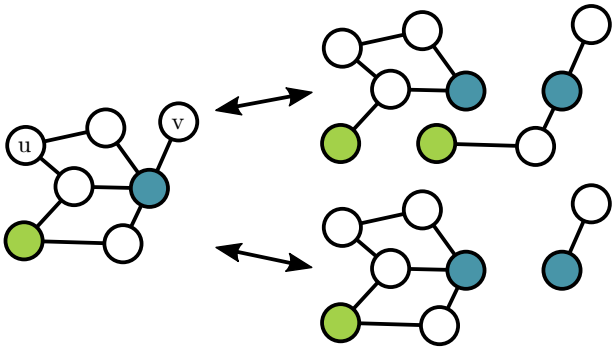


Figure 1 Consider the graph on the left, which can be decomposed into colored 3-bounded disc types (which are rooted at u and v in this example) in more than one way. However, it is always possible to recover the original graph by identifying vertices of the same color. Furthermore, every mapping is bijective because every color is assigned at most once per disc. If the colored vertices correspond to the vertices in V_α, every forbidden graph $F \in \mathcal{F}_n$ from Theorem 10 corresponds to a decomposition into edge-disjoint colored q-bounded discs $F' \in \mathcal{F}'_n$ in Theorem 17, which intersect only at colored vertices.

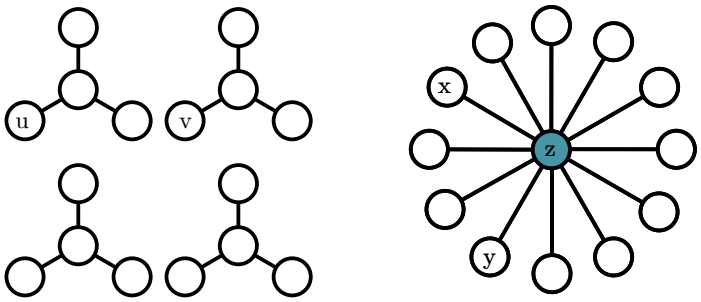


Figure 2 The above graph, which is composed of 3-stars and a $\omega(1)$-star with root z and which should be thought of as a subgraph of some larger graph, illustrates the need for colors in our analysis of the streaming property tester. Although the 2-bounded discs of u, v x and y are all 3-stars (with constant probability over the randomness of the neighbor queries), exploring u and v by q-RBFS does not result in finding a 6-star, while it is likely to find a 6-star by exploring x and y. Even if we prove that the probability that a q-SC finds uncolored 3-stars is lower bounded by some constant fraction of the probability that q-RBFS finds uncolored 3-stars, we still cannot rule out that q-SC might tend to find leaves of the small stars (like u and v) while q-RBFS tends to find leaves of the big star (like x and y). Observe that here, z is the only vertex that is likely contained in two different q-RBFS due to its high degree.

B Missing Pseudocodes from Section 3 and 4

The pseudocodes for the q-random BFS and for collecting a q-bounded disc from a vertex in stream are given below.

Algorithm 1 q-random BFS.

function RandomBFS(G, v, q)
  $Q \leftarrow$ empty queue; enqueue(Q, v)
  $\forall w \in V : \ell[w] \leftarrow \infty$
  $\ell[v] \leftarrow 0$
  $H \leftarrow (\{v\}, \emptyset)$ with v as root
  while Q not empty **do**
    $u \leftarrow$ pop element from Q
    for $1 \leq i \leq q$ **do**
      $s_{u,i} \leftarrow$ query oracle for random neighbor of u
      add vertex $s_{u,i}$ and edge $(u, s_{u,i})$ to H
      if $(\ell[u] < q - 1) \wedge (\ell[s_{u,i}] = \infty)$ **then**
        $\ell[s_{u,i}] \leftarrow \ell[u] + 1$
        enqueue$(Q, s_{u,i})$
  return undirected H without parallel edges
end function

Algorithm 2 Collecting a q-bounded disc from a vertex in stream.

function StreamCollect$(\mathcal{S}(G), v, q)$
  $U \leftarrow \{v\}$
  $\forall u \in V : d_u \leftarrow 0, \ell_u \leftarrow \infty$
  $\ell_v \leftarrow 0; F \leftarrow \emptyset$
  $H = (U, F)$ with v marked as root
  for $(u, w) \leftarrow$ next edge in the stream **do**
    if $(\{u, w\} \cap U \neq \emptyset)$ **then**
      if $(u \in U \Rightarrow (\ell_u < q \wedge d_u < q^{2q}) \vee (w \in U \Rightarrow (\ell_w < q \wedge d_w < q^{2q}))$ **then**
        $U \leftarrow U \cup \{u, w\}$
        $F \leftarrow F \cup (u, w)$
        $d_u \leftarrow d_u + 1; d_w \leftarrow d_w + 1$
        $\ell_u \leftarrow \min(\ell_u, \ell_w + 1); \ell_w \leftarrow \min(\ell_w, \ell_u + 1)$
  return H
end function

C Missing Proofs from Section 5

Proof of Theorem 4. We let $q_0 = q_0(\varepsilon)$ denote the query complexity of Π. Let $n = |V|$. We present our testing algorithm. Let $q = c \cdot q_0$ for some constant c from Theorem 10. Let $\alpha = \frac{c_*(q)^4 \delta^8}{10^9 |\mathcal{H}_q|^2 q^{2q} c'_q}$, where $c'_q = \sum_{i=0}^{q+1} q^{2qi}$, and $\delta = \frac{1}{200|\mathcal{H}_q|}$. If $n \leq n_0 := \frac{qc_q}{\alpha^2}$, then we simply store the whole graph. If $n > n_0$, we proceed as follows. Let $\mathcal{F}_n$ be the set of forbidden subgraphs that characterize Π as stated in Theorem 10. We sample $s \geq \max\{\frac{1}{20\sqrt{\alpha q^{2q} \cdot c'_q}}, \frac{5000|\mathcal{H}_q|}{c_*(q)\delta^3}\}$ vertices $S \subseteq V$ and run StreamCollect$(\mathcal{S}(G), v, q)$ for each $v \in S$ to obtain a subgraph $H_v = (V_v, E_v)$ of G. If $H = \cup_{v \in S} H_v$ contains a forbidden subgraph $F \in \mathcal{F}_n$, the tester rejects, otherwise it accepts. See Algorithm 3 for details.

Algorithm 3 Testing graph property Π in random order stream.

function StreamTest($\mathcal{S}(G), n, \varepsilon, \mathcal{F}_n$)
  $S \leftarrow$ sample s vertices u.a.r. from V
  for all $v \in S$ **do**
    $H_v \leftarrow (V_v, E_v) =$ StreamCollect($\mathcal{S}(G), v, q$)
  $H \leftarrow (\cup_v V_v, \cup_v E_v)$
  if there exists $F \in \mathcal{F}_n$ such that H contains a subgraph F **then**
    Output **Reject**
  else
    Output **Accept**
end function

The space complexity of the algorithm is $s \cdot q_0^{O(q_0)} = O_{q_0}(1)$ words. For the correctness of the algorithm, we note that for any property Π that is constant-query testable with one-sided error, then with probability 1, we will not see any $F \in \mathcal{F}'_n$ if the graph G satisfies Π.

On the other hand, if G is ε-far from satisfying Π, then by Theorem 17, with probability at least $\frac{2}{3}$, the subgraph S_q spanned by the union of q-bounded discs rooted at q uniformly sampled vertices from G will span a subgraph that is isomorphic to some $F \in \mathcal{F}'_n$. Note that, in contrast to the algorithm above, the analysis uses the decomposition of forbidden subgraphs in $\mathcal{F}_n$ into colored q-discs given by Theorem 17. The key idea is to use the q-bounded discs that StreamCollect collects and the implicit colors (which are not observed by StreamCollect, but can be used in the analysis to identify vertices in V_α) to stitch forbidden subgraphs from $\mathcal{F}'_n$ that are discovered by RandomBFS. We prove that with sufficient probability, for each colored q-bounded disc Δ, StreamCollect finds at least as many copies of Δ as RandomBFS, and therefore, it can reproduce the same types of forbidden subgraphs from $\mathcal{F}'_n$.

By Markov's inequality and the union bound, the probability that at least one q-RBFS in the canonical tester for Π will return a colored q-bounded disc that is isomorphic to a disc Δ' such that $\mathrm{Reach}_G(\Delta') < 2\delta = \frac{1}{100|\mathcal{H}_q|}$ is at most $\frac{1}{100}$. Let $\mathcal{D}$ be the set of all colored q-bounded discs Δ such that $\mathrm{Reach}_G(\Delta) \geq 2\delta$.

By Lemma 21, with probability at least $1 - \frac{1}{100}$, for every $\Delta \in \mathcal{D}$, the number of graphs H_v obtained by StreamCollect that contain a subgraph isomorphic to Δ is at least $100|\mathcal{H}_q| \cdot \mathrm{Reach}_G(\Delta) \geq 1$. By (implicitly) coloring all vertices in V_α, it follows from Theorem 17 that H contains a forbidden subgraph from $\mathcal{F}'_n$ with probability $1 - \frac{1}{100} - \frac{1}{100} > \frac{2}{3}$. ◀

Multicriteria Cuts and Size-Constrained k-Cuts in Hypergraphs

Calvin Beideman
University of Illinois, Urbana-Champaign, IL, USA
calvinb2@illinois.edu

Karthekeyan Chandrasekaran
University of Illinois, Urbana-Champaign, IL, USA
karthe@illinois.edu

Chao Xu
The Voleon Group, Berkeley, CA, USA
the.chao.xu@gmail.com

Abstract

We address counting and optimization variants of multicriteria global min-cut and size-constrained min-k-cut in hypergraphs.

1. For an r-rank n-vertex hypergraph endowed with t hyperedge-cost functions, we show that the number of multiobjective min-cuts is $O(r2^{tr}n^{3t-1})$. In particular, this shows that the number of parametric min-cuts in constant rank hypergraphs for a constant number of criteria is strongly polynomial, thus resolving an open question by Aissi, Mahjoub, McCormick, and Queyranne [1]. In addition, we give randomized algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in strongly polynomial-time.
2. We also address node-budgeted multiobjective min-cuts: For an n-vertex hypergraph endowed with t vertex-weight functions, we show that the number of node-budgeted multiobjective min-cuts is $O(r2^r n^{t+2})$, where r is the rank of the hypergraph, and the number of node-budgeted b-multiobjective min-cuts for a fixed budget-vector $b \in \mathbb{R}^t_+$ is $O(n^2)$.
3. We show that min-k-cut in hypergraphs subject to constant lower bounds on part sizes is solvable in polynomial-time for constant k, thus resolving an open problem posed by Queyranne [11]. Our technique also shows that the number of optimal solutions is polynomial.

All of our results build on the random contraction approach of Karger [12]. Our techniques illustrate the versatility of the random contraction approach to address counting and algorithmic problems concerning multiobjective min-cuts and size-constrained k-cuts in hypergraphs.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization

Keywords and phrases Multiobjective Optimization, Hypergraph min-cut, Hypergraph-k-cut

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.17

Category RANDOM

Related Version http://arxiv.org/abs/2006.11589

Funding *Calvin Beideman*: Supported in part by NSF CCF-1907937.
Karthekeyan Chandrasekaran: Supported in part by NSF CCF-1907937 and CCF-1814613.

1 Introduction

Cuts and partitioning play a central role in combinatorial optimization and have numerous theoretical as well as practical applications. We consider multicriteria cut problems in hypergraphs. Let $G = (V, E)$ be an n-vertex hypergraph and $c_1, \ldots, c_t : E \to \mathbb{Z}_+$ be t non-negative hyperedge-cost functions, where t is a constant. The cost of a subset F of hyperedges under criterion $i \in [t]$ is $c_i(F) := \sum_{e \in F} c_i(e)$. For a positive integer k, a subset


© Calvin Beideman, Karthekeyan Chandrasekaran, and Chao Xu;
licensed under Creative Commons License CC-BY



of hyperedges that crosses a k-partition $(U_1, \ldots, U_k)$ of the vertex set is said to be a k-cut. We refer to a 2-cut simply as a cut. We recall that the rank of a hypergraph G is the size of the largest hyperedge in G (the rank of a graph is 2).

Since we have several criteria, there may not be a single cut that is best for all criteria. In multicriteria optimization, there are three important notions to measure the quality of a cut: (i) parametric min-cuts, (ii) pareto-optimal cuts, and (iii) multiobjective min-cuts. A cut F is a *parametric min-cut* if there exist positive multipliers $\mu_1, \ldots, \mu_t \in \mathbb{R}_+$ such that F is a min-cut in the hypergraph G with hyperedge costs given by $c_\mu(e) := \sum_{i=1}^{t} \mu_i c_i(e)$ for all $e \in E$. A cut F *dominates* another cut F' if $c_i(F) \leq c_i(F')$ for every $i \in [t]$ and there exists $i \in [t]$ such that $c_i(F) < c_i(F')$. A cut F is *pareto-optimal* if it is not dominated by any other cut. For a budget-vector $b \in \mathbb{R}_+^{t-1}$, a cut F is a *b-multiobjective min-cut* if $c_i(F) \leq b_i$ for every $i \in [t-1]$ and $c_t(F)$ is minimum subject to these constraints. A cut F is a *multiobjective min-cut* if there exists a non-negative budget-vector $b \in \mathbb{R}_+^{t-1}$ for which F is a b-multiobjective min-cut. These three notions satisfy the following relationship with the containment being possibly strict (see the full version of this paper for a proof [4]):

$$\text{Parametric min-cuts} \subseteq \text{Pareto-optimal cuts} \subseteq \text{Multiobjective min-cuts}. \tag{1}$$

There is also a natural notion of min-cuts under node-weighted budget constraints. Let $w_1, \ldots, w_t : V \to \mathbb{R}_+$ be vertex-weight functions and $c : E \to \mathbb{R}_+$ be a hyperedge-cost function. For a budget-vector $b \in \mathbb{R}_+^t$, a subset $F \subseteq E$ of hyperedges is a *node-budgeted b-multiobjective min-cut* if $F = \delta(U)$ for some subset $\emptyset \neq U \subsetneq V$ with $\sum_{u \in U} w_i(u) \leq b_i$ for all $i \in [t]$ and $c(F)$ is minimum among all such subsets of E. A cut F is a *node-budgeted multiobjective min-cut* if there exists a non-negative budget-vector b for which F is a node-budgeted b-multiobjective min-cut. In this work, we address the following natural questions concerning multiobjective min-cuts and min-k-cuts:

1. Multiobjective min-cuts: Is the number of multiobjective min-cuts at most strongly polynomial?
2. Node-budgeted multiobjective min-cuts: Is the number of node-budgeted multiobjective min-cuts at most strongly polynomial?
3. Size-constrained min-k-cut: For fixed positive integers k and $s_1, \ldots, s_k$ (all constants), a vertex-weight function $w : V \to \mathbb{Z}_+$, and a hyperedge-cost function $c : E \to \mathbb{R}_+$, can we compute a k-cut F with minimum $c(F)$ subject to the constraint that F is the set of hyperedges crossing some k-partition $(U_1, \ldots, U_k)$ of V where $\sum_{u \in U_i} w(u) \geq s_i$ for every $i \in [k]$ in polynomial-time? Is the number of optimal solutions strongly polynomial?

Previous Work. For single criterion, a classic result of Dinitz, Karzanov, and Lomonosov [7] shows that the number of min-cuts in an n-vertex graph is $O(n^2)$ (also see Karger [12]). The same upper bound was shown to hold for constant-rank hypergraphs by Kogan and Krauthgamer [14] and for arbitrary-rank hypergraphs by Chekuri and Xu [6] and by Ghaffari, Karger, and Panigrahi [8] via completely different techniques. For $t = 2$ criteria in graphs, Mulmuley [15] showed an $O(n^{19})$ upper bound on the number of parametric min-cuts. For t criteria in constant-rank hypergraphs, Aissi, Mahjoub, McCormick, and Queyranne [1] showed that the number of parametric min-cuts is $\tilde{O}(m^t n^2)$, where m is the number of hyperedges, using the fact that the number of approximate min-cuts in constant-rank hypergraphs is polynomial. Karger [13] improved this bound to $O(n^{t+1})$ by a clever and subtle argument based on his random contraction algorithm; we will describe his argument later. Karger also constructed a graph that exhibited $\Omega(n^{t/2})$ parametric min-cuts.

Armon and Zwick [3] showed that all pareto-optimal cuts in graphs can be enumerated in pseudo-polynomial time. For $t = 2$ criteria in constant-rank hypergraphs, Aissi et al [1] showed an upper bound of $\tilde{O}(n^5)$ on the number of pareto-optimal cuts – this was the first result showing a strongly polynomial upper bound. Aissi et al raised the question of whether the number of pareto-optimal cuts is strongly polynomial for a constant number t of criteria in constant-rank hypergraphs (or even in graphs). Note that, by containment relationship (1), answering our first question affirmatively would also answer their open question. On a related note, Aissi, Mahjoub, and Ravi [2] designed a random contraction based algorithm to solve the b-multiobjective min-cut problem in graphs. The correctness analysis of their algorithm also implies that the number of b-multiobjective min-cuts in graphs for a fixed budget-vector $b \in \mathbb{R}_+^{t-1}$ is $O(n^{2t})$. We emphasize the subtle, but important, distinction between the number of b-multiobjective min-cuts for a fixed budget-vector b and the number of multiobjective min-cuts.

Node-budgeted multiobjective min-cut has a rich literature extending nicely to submodular functions. For graphs, Armon and Zwick [3] gave a polynomial-time algorithm to find a minimum valued cut with at most b vertices in the smaller side. Goemans and Soto [9] addressed the more general problem of minimizing a symmetric submodular function $f : 2^V \to \mathbb{R}$ over a downward-closed family $\mathcal{I}$. Recall that the hypergraph cut function is symmetric submodular and the family of vertex subsets satisfying node-weighted budget constraints is in fact downward-closed. Goemans and Soto extended Queyranne's submodular minimization algorithm to enumerate all the $O(n)$ minimal minimizers in $\mathcal{I}$ using $O(n^3)$ oracle calls to the function f and the family $\mathcal{I}$. Their result implies that the number of minimal minimizers is $O(n)$, but it is straightforward to see that the total number of minimizers could be exponential. For the special case of node-budgeted multiobjective min-cuts in graphs, Aissi, Mahjoub, and Ravi [2] gave a faster algorithm than that of Goemans and Soto – their algorithm is based on random contraction, runs in $\tilde{O}(n^2)$-time, and shows that the number node-budgeted b-multiobjective min-cuts in graphs for a fixed budget-vector $b \in \mathbb{R}_+^t$ is $O(n^2)$.

For size-constrained min-k-cut, if we allow arbitrary sizes (i.e., arbitrary lower bounds), then the problem becomes NP-hard even for $k = 2$ as it captures the well-studied min-bisection problem in graphs. If we consider constant sizes but arbitrary k, then the problem is again NP-hard in graphs [10]. So, our focus is on constant k and constant sizes. Guinez and Queyranne [11] raised size-constrained min-k-cut with unit vertex-weights as a subproblem towards resolving the complexity of the submodular k-partitioning problem. In submodular k-partitioning, we are given a submodular function $f : 2^V \to \mathbb{R}$ (by value oracle) and a fixed constant integer k (e.g., $k = 2, 3, 4, 5, \ldots$) and the goal is to find a k-partition $(U_1, \ldots, U_k)$ of the ground set V so as to minimize $\sum_{i=1}^{k} f(U_i)$. The complexity of even special cases of this problem are open: e.g., if the submodular function f is the cut function of a given hypergraph, then its complexity is unknown.[1] Guinez and Queyranne showed surprisingly strong non-crossing properties between optimum solutions to size-constrained $(k-1)$-partitioning (constant size lower bounds on the parts) and optimum solutions to k-partitioning. This motivated them to study the size-constrained min-k-cut problem in hypergraphs for unit vertex-weights as a special case. They showed that size-constrained min-k-cut for unit vertex-weights is solvable in polynomial-time in constant-rank hypergraphs

[1] We note that if the submodular function f is the cut function of a given hypergraph, then the submodular k-partition problem is not identical to hypergraph k-cut as the two objectives are different. However, if the submodular function is the cut function of a given graph, then the submodular k-partition problem coincides with the graph k-cut problem which is solvable in polynomial-time.

(with exponential run-time dependence on the rank) and mention the open problem of designing an algorithm for it in arbitrary-rank hypergraphs. The size-constrained min-k-cut problem for unit sizes (i.e., all size lower-bounds $s_1, \ldots, s_k$ are equal to one) is known as the hypergraph k-cut problem. The hypergraph k-cut problem was shown to admit a polynomial-time algorithm only recently [5] via a non-uniform random contraction algorithm.

1.1 Our Contributions

Our high-level contribution is in showing the versatility of the random contraction technique to address algorithmic and counting problems concerning multiobjective min-cuts and size-constrained min-k-cuts in hypergraphs. All of our results build on the random contraction technique with additional insights.

Our first result is a strongly polynomial upper bound on the number of multiobjective min-cuts in constant-rank hypergraphs.

▶ **Theorem 1.1.** *The number of multiobjective min-cuts in an r-rank, n-vertex hypergraph G with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$.*

We emphasize that our upper bound is over all possible non-negative budget-vectors (in contrast to the number of b-multiobjective min-cuts for a fixed budget-vector b). Theorem 1.1 and Proposition 1 imply that the number of pareto-optimal cuts in constant-rank hypergraphs is $O(n^{3t-1})$ and hence, is strongly polynomial for constant number of criteria. This answers the main open question posed by Aissi, Mahjoub, McCormick, and Queyranne [1]. We also design randomized polynomial time algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in constant-rank hypergraphs (see Section A.3). Independent of our work, Rico Zenklusen has also shown Theorem 1.1. We learned after submission of this work that his approach leads to *deterministic* polynomial time algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts in constant-rank hypergraphs.

Given the upper bound in Theorem 1.1, a discussion on the lower bound is in order. We recall that Karger [13] constructed a graph with t edge-cost functions that exhibited $\Omega(n^{t/2})$ parametric min-cuts. This is also a lower bound on the number of pareto-optimal cuts and multiobjective min-cuts by (1). We improve this lower bound for pareto-optimal cuts by constructing a graph with t edge-cost functions that exhibits $\Omega(n^t)$ pareto-optimal cuts (see Section A.4). Our instance also exhibits the same lower bound on the number of b-multiobjective min-cuts for a fixed budget-vector b.

Our next result is an upper bound on the number of node-budgeted multiobjective min-cuts and node-budgeted b-multiobjective min-cuts.

▶ **Theorem 1.2.** **1.** *The number of node-budgeted multiobjective min-cuts in an r-rank, n-vertex hypergraph with t vertex-weight functions is $O(r2^r n^{t+2})$.*

2. *For a fixed budget-vector $b \in \mathbb{R}^t_+$, the number of node-budgeted b-multiobjective min-cuts in an n-vertex hypergraph with t vertex-weight functions is $O(n^2)$.*

We draw the reader's attention to the distinction between the two parts in Theorem 1.2. The first part implies that the number of node-budgeted multiobjective min-cuts is strongly polynomial in constant-rank hypergraphs for constant number of vertex-weight functions. The second part implies that the number of node-budgeted b-multiobjective min-cuts for any fixed budget-vector $b \in \mathbb{R}^t_+$ is strongly polynomial in arbitrary-rank hypergraphs for any number t of vertex-weight functions.

Our final result shows that the size-constrained min-k-cut problem can be solved in polynomial time for constant k and constant sizes (in arbitrary-rank hypergraphs).

▶ **Theorem 1.3.** *Let $k \geq 2$ be a fixed positive integer and let $1 \leq s_1 \leq s_2 \leq \ldots \leq s_k$ be fixed positive integers. Let G be an n-vertex hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$. Then, there exists a polynomial-time algorithm that takes (G, c) as input and returns a fixed w-weighted s-size-constrained min-k-cut for any choice of vertex-weight function $w : V \to \mathbb{Z}_+$ with probability $\Omega\left(\frac{1}{n^{2\sigma_{k-1}+1}}\right)$, where $\sigma_{k-1} := \sum_{i=1}^{k-1} s_i$.*

Theorem 1.3 resolves an open problem posed by Guinez and Queyranne [11]. A structural consequence of Theorem 1.3 is that the number of size-constrained min-k-cuts (over all possible node-weight functions $w : V \to \mathbb{Z}_+$) in a given hypergraph is polynomial for constant sizes and constant k.

We refer the reader to Table 1 for a comparison of known results and our contributions.

Table 1 Text in gray refers to known results while text in black illustrates results from this work. Here, t denotes the number of criteria (i.e., the number of hyperedge-cost/vertex-weight functions), r denotes the rank of the hypergraph, and n denotes the number of vertices.

Problem	**Graphs**	**r-rank Hypergraphs**	**Hypergraphs**
# of Parametric Min-Cuts	$O(n^{t+1})$ [13] $\Omega(n^{t/2})$ [13]	$O(2^r n^{t+1})$ [13]	OPEN
# of Pareto-Optimal Cuts	$\tilde{O}(n^5)$ for $t = 2$ [1] $O(n^{3t-1})$ [Thm 1.1] $\Omega(n^t)$ [Thm A.2]	$O(r2^{rt}n^{3t-1})$ [Thm 1.1]	OPEN
# of b-Multiobjective Min-Cuts	$O(n^{2t})$ [2] $\Omega(n^t)$ [Thm A.2]	$O(r2^{rt}n^{2t})$ [Thm 2.1]	OPEN
# of Multiobjective Min-Cuts	$O(n^{3t-1})$ [Thm 1.1] $\Omega(n^t)$ [Thm A.2]	$O(r2^{rt}n^{3t-1})$ [Thm 1.1]	OPEN
# of Node-Budgeted b-Multiobjective Min-Cuts	$O(n^2)$ [2]	$O(n^2)$ [Thm 1.2]	$O(n^2)$ [Thm 1.2]
# of Node-Budgeted Multiobjective Min-Cuts	$O(n^{t+2})$ [Thm 1.2]	$O(r2^r n^{t+2})$ [Thm 1.2]	OPEN
Node-Weighted s-Size-Constrained k-cut (const. k and const. $s \in \mathbb{Z}^k$)	Poly-time [11]	Poly-time [11]	Poly-time [Thm 1.3]

1.2 Technical Overview

As mentioned earlier, all our results build on the random contraction technique introduced by Karger [12] to solve the global min-cut problem in graphs. Here, a *uniform* random edge of the graph is contracted in each step until the graph has only two nodes; the set of edges between the two nodes is returned as the cut. Karger showed that this algorithm returns a fixed global min-cut with probability $\Omega(n^{-2})$. As a consequence, the number of min-cuts in an n-vertex graph is $O(n^2)$. The algorithm extends naturally to r-rank hypergraphs, however the naive analysis will only show that the algorithm returns a fixed global min-cut with probability $\Omega(n^{-r})$. Kogan and Krauthgamer [14] introduced an LP-based analysis thereby showing that the algorithm indeed succeeds with probability $\Omega(2^{-r}n^{-2})$. As a consequence, the number of global min-cuts in constant-rank hypergraphs is also $O(n^2)$.

In a recent work, Karger observed that uniform random contraction can also be used to bound the number of parametric min-cuts in constant-rank hypergraphs. We describe his argument for graphs since two of our theorems build on it. Suppose we fix the multipliers

$\mu_1, \ldots, \mu_t$ in the parametric min-cut problem, then a fixed min c_μ-cost cut can be obtained with probability $\Omega(n^{-2})$ by running the random contraction algorithm with respect to the edge-cost function c_μ. Karger suggested an alternative viewpoint of the execution of the algorithm for the edge-cost function c_μ. For simplicity, we assume parallel edges instead of costs, i.e., $c_i(e) \in \{0, 1\}$ for every edge e and every criterion $i \in [t]$. Let E_i be the set of edges with non-zero weight in the i'th criterion. The execution of the random-contraction algorithm wrt c_μ can alternatively be specified as follows: a permutation π_i of the edges in E_i for each $i \in [t]$ and an *interleaving* indicating at each step whether the algorithm contracts the next edge from π_1 or π_2 or ... or π_t. Critically, the sequences π_i for every $i \in [t]$ can be assumed to be uniformly random. Thus, we can move all randomness upfront, namely pick a uniform random permutation π_i for each criterion $i \in [t]$. Now, instead of returning one cut, we return the collection of cuts produced by contracting along all possible interleavings. This modified algorithm no longer depends on the specific multipliers $\mu_1, \ldots, \mu_t$ and hence, a parametric min-cut for any fixed choice of multipliers $\mu_1, \ldots, \mu_t$ will be in the output collection with probability at least $\Omega(n^{-2})$. It remains to bound the number of interleavings since that determines the number of cuts in the returned collection: the crucial observation here is that the number of interesting interleavings is only n^{t-1}. This is because interleaved contractions produce the same final graph as performing a certain number of contractions according to π_1 (until the number of vertices is n_1 say), then a certain number of contractions based on π_2 (until the number of vertices is n_2 say), and so on. So, the order of contractions becomes irrelevant and only the number of vertices $n_1, \ldots, n_t$ are relevant. Overall, this implies that the number of parametric min-cuts is $O(n^{t+1})$. We emphasize that this *interleaving argument* relies crucially on the basic random contraction algorithm picking edges to contract according to a *uniform distribution* (allowing the permutations $\pi_1, \ldots, \pi_t$ to be uniform random permutations).

Next, we describe our approach underlying the proof of Theorem 1.1, but for graphs. In order to bound the number of multiobjective min-cuts through the interleaving argument, we first need a random-contraction based algorithm to solve the b-multiobjective min-cut problem. Indeed, Aissi, Mahjoub, and Ravi [2] designed a random-contraction based algorithm to solve the b-multiobjective min-cut problem in graphs. Their algorithm proceeds as follows: Let $U_0 := \emptyset$ and for each $i \in [t-1]$, let U_i be the set of vertices $u \in V - \cup_{j=1}^{i-1} U_j$ for which $c_i(\delta(u)) > b_i$ (known as the set of i-infeasible vertices), and let $U_t := V - \cup_{j=1}^{t-1} U_j$. In each step, they pick $i \in [t]$ with probability proportional to the number of i-infeasible vertices (i.e., $|U_i|$) and pick a random edge e among the ones incident to U_i with probability proportional to $c_i(e)$, contract e, and repeat. Unfortunately, this algorithm does not have the *uniform distribution* that is crucially necessary to apply Karger's interleaving argument. To introduce uniformity to the distribution, we modify this algorithm in two ways:

1. At each step we deterministically choose the criterion i corresponding to the largest U_i (as opposed to picking i randomly with probability proportional to $|U_i|$).
2. Next, we choose a uniform random edge e from among all edges in the graph with probability proportional to $c_i(e)$ (as opposed to picking an edge only from among the edges incident to U_i). We contract this chosen edge e.

These two features bring a *uniform distribution* property to the algorithm, which in turn, allows us to apply the interleaving argument. With these two features, we show that the algorithm returns a fixed b-multiobjective min-cut for a fixed budget-vector b with probability $\Omega(n^{-2t})$. Armed with the two features, we move all randomness upfront using the interleaving argument. As a consequence, we obtain that the total number of multiobjective min-cuts (irrespective of the choice of budget-vector b) is $O(n^{3t-1})$. For constant-rank hypergraphs, we

perform an LP-based analysis of our algorithm for b-multiobjective min-cut (thus, extending Kogan and Krauthgamer's analysis) to arrive at the same success probability. The interleaving argument for constant-rank hypergraphs proceeds similarly.

We emphasize that the interleaving argument does not extend to arbitrary-rank hypergraphs. This is because, the random contraction based algorithm that we know for arbitrary-rank hypergraphs crucially requires non-uniform contractions (the next hyperedge to contract is chosen from a distribution that depends on the current sizes of all hyperedges), so we cannot assume that the permutations $\pi_1, \pi_2, \ldots, \pi_t$ are uniformly random. Consequently, we do not even know if the number of parametric min-cuts in a hypergraph is at most strongly polynomial. Another interesting open question here is whether the b-multiobjective min-cut problem in hypergraphs is solvable in polynomial-time even for $t = 2$ criteria. We have arrived at hypergraph instances (with large rank) for which Aissi, Mahjoub, and Ravi's approach (as well as our modified approach) will never succeed, even with non-uniform random contractions.

Next, we outline the proof of Theorem 1.2. The approach is to again design a random-contraction algorithm that returns a fixed node-budgeted b-multiobjective min-cut with probability $\Omega(n^{-2})$ (in both constant-rank and arbitrary-rank hypergraphs). Such an algorithm would imply the second part of the theorem immediately while the first part would follow if we can apply an interleaving-like argument (i.e., the designed algorithm performs uniform random contractions). Our approach is essentially an extension of the approach by Goemans and Soto who suggested contracting the infeasible vertices together (a vertex u is infeasible if $w_i(u) > b_i$ for some $i \in [t]$. Aissi, Mahjoub, and Ravi show that doing this additional step after each random contraction step returns a fixed node-budgeted b-multibjective min-cut with probability $\Omega(n^{-2})$ in graphs. Our main contribution is showing that this additional "contracting infeasible vertices together" step in conjunction with (1) uniform random contractions for constant-rank hypergraphs and (2) non-uniform random contractions for arbitrary hypergraphs succeeds with the required probability. Next, a naive interleaving-like argument can be applied for constant-rank hypergraphs to conclude that the number of node-budgeted multiobjective min-cuts is $O(n^{t+3})$. We improve this to $O(n^{t+2})$ with a more careful argument.

Finally, we outline our approach for Theorem 1.3. Guinez and Queyranne address size-constrained k-cut in constant-rank hypergraphs for unit vertex-weights by performing uniform random contractions until the number of nodes in the hypergraph is close to $\sum_{i=1}^{k} s_i$ at which point they return a uniform random cut. Their success probability has exponential dependence on the rank. The key technical ingredient to bring down the exponential dependence on rank is the use of non-uniform contractions. For the special case of unit sizes and unit vertex-weights (i.e., the hypergraph k-cut problem), Chandrasekaran, Xu, and Yu [5] introduced an explicit non-uniform distribution that leads to a success probability of $\Omega(n^{-2(k-1)})$. Our algorithm extends the non-uniform distribution to arbitrary but constant sizes (as opposed to just unit sizes), yet without depending on vertex-weights. Our analysis takes care of the vertex-weight function through weight tracking, i.e., by declaring the weight of a contracted node to be the sum of the weight of the vertices in the hyperedge being contracted. We note that our algorithm's success probability when specialized to the case of unit vertex-weights and unit sizes is weaker than the success probability of the algorithm by Chandrasekaran, Xu, and Yu (by a factor of n). We leave it as an open question to improve this. On the other hand, our algorithm has the added advantage that it does not even take the vertex-weight function w as input and yet succeeds in returning a w-vertex-weighted s-size-constrained k-cut for any choice of w with inverse polynomial probability.

Organization. In Section 2, we bound the number of multiobjective min-cuts (and prove Theorem 1.1). Due to page limit, we give efficient algorithms to enumerate all multiobjective min-cuts and all pareto-optimal cuts, and present lower bounds on the number of pareto-optimal cuts and the number of b-multiobjective min-cuts in the appendix. In Section 3, we give an algorithm to solve size-constrained min-k-cut, thereby proving Theorem 1.3. Due to page limit, we defer the details of node-budgeted multiobjective min-cuts and a proof of Theorem 1.2 to the full version of the paper [4].

1.3 Preliminaries

We define the random contraction procedure that is central to all of our algorithms. Let $G = (V, E)$ be a hypergraph, and let $U \subseteq V$ be a set of vertices in G. We define G *contract* U, denoted G/U, to be a hypergraph on the vertex set $(V \setminus U) \cup \{u\}$, where u is a newly introduced vertex. The hyperedges of G/U are defined as follows. For each hyperedge $e \in E$ of G, such that $e \not\subseteq U$, G/U has a corresponding hyperedge e', where $e' := e$ if $e \subseteq V \setminus U$ and $e' := (e \setminus U) \cup \{u\}$ otherwise. If w is a vertex-weight function for G, then we will also use w as a vertex-weight function for G/U. We define the weight of the newly introduced vertex u as $w(u) := \sum_{v \in U} w(v)$.

We will need the following lemma that will be used in the analysis of two of our algorithms. We refer the reader to the full version of this paper for the proof.

▶ **Lemma 1.4.** *Let r, γ, n be positive integers with $n \geq \gamma \geq r + 1 > 2$. Let $f : \mathbb{N} \to \mathbb{R}_+$ be a positive-valued function defined over natural numbers. Then, the optimum value of the linear program (LP_1) defined below is* $\min_{2 \leq j \leq r}(1 - \frac{j}{\gamma - r + j}) f(n - j + 1)$.

$$\underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\textit{minimize}} \quad \sum_{j=2}^{r} (x_j - y_j) f(n - j + 1) \tag{LP_1}$$

$$\textit{subject to} \quad 0 \leq y_j \leq x_j \ \forall j \in \{2, \ldots, r\} \tag{2}$$

$$\sum_{j=2}^{r} x_j = 1 \tag{3}$$

$$\gamma \sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j \tag{4}$$

2 Multiobjective Min-Cuts and Pareto-Optimal Cuts

In this section, we give upper and lower bounds on the number of multiobjective min-cuts and pareto-optimal cuts and prove Theorem 1.1.

Let $G = (V, E)$ be an r-rank hypergraph and let $c_1, \ldots, c_t : E \to \mathbb{R}_+$ be cost functions on the hyperedges of G. For a subset F of hyperedges, we will use $c_i(F)$ to denote $\sum_{e \in F} c_i(e)$. For a subset U of vertices, we will use $\overline{U}$ to denote $V \setminus U$ and $\delta(U)$ to denote the set of hyperedges that intersect both U and $\overline{U}$. For a vertex v, we will use $\delta(v)$ to denote $\delta(\{v\})$. A subset F of hyperedges is a cut if there exists a partition $(U, \overline{U})$ such that $F = \delta(U)$. We refer the reader to Section 1 for the definitions of b-multiobjective min-cuts, multiobjective min-cuts, and pareto-optimal cuts.

We begin with a randomized algorithm for the b-multiobjective min-cut problem in Section 2.1. We take an alternative viewpoint of this randomized algorithm in Section 2.2 to prove Theorem 1.1. Since all pareto-optimal cuts are multiobjective min-cuts, Theorem 1.1

also implies that the number of pareto-optimal cuts in an r-rank n-vertex hypergraph G with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$. In Section A.3, we give randomized polynomial-time algorithms to enumerate all pareto-optimal cuts and all multiobjective min-cuts. In Section A.4, we show a lower bound of $\Omega(n^t)$ on the number of pareto-optimal cuts and on the number of b-multiobjective min-cuts.

2.1 Finding b-Multiobjective Min-Cuts

In this section, we design a randomized algorithm for the b-multiobjective min-cut problem, which is formally defined below. We use Algorithm 1. We summarize its correctness and run-time guarantees in Theorem 2.1.

b-Multiobjective Min-Cut
Given: A hypergraph $G = (V, E)$ with hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$ and a budget-vector $b \in \mathbb{R}_+^{t-1}$.
Goal: A b-multiobjective min-cut.

Algorithm 1 b-Multiobjective Min-Cut.

b-Multiobjective-Min-Cut(G, r, t, c, b):

Input: An r-rank hypergraph $G = (V, E)$, hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$ and a budget-vector $b \in \mathbb{R}_+^{t-1}$.

If $|V| \leq rt$:
  $X \leftarrow$ a random subset of V
  return $\delta(X)$
For $i = 1, \ldots, t-1$:
  $U_i \leftarrow \{v \in V \setminus (\bigcup_{j=1}^{i-1} U_j) \colon c_i(\delta(v)) > b_i\}$
$U_t \leftarrow V \setminus \bigcup_{j=1}^{t-1} U_j$
$i \leftarrow \arg\max_{j \in [t]} |U_j|$
$e \leftarrow$ a random hyperedge chosen according to $\Pr[e = e'] = \frac{c_i(e')}{c_i(E)}$
$G' \leftarrow G/e$
Return b-Multiobjective-Min-Cut(G', r, t, c, b)

▶ **Theorem 2.1.** *Let $G = (V, E)$ be an r-rank n-vertex hypergraph with hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$ and let $b \in \mathbb{R}_+^{t-1}$ be a budget-vector. Let F be an arbitrary b-multiobjective min-cut. Then, Algorithm 1 outputs F with probability at least Q_n, where*

$$Q_n := \begin{cases} \frac{1}{2^{rt}} & \text{if } n \leq rt, \text{ and} \\ \frac{2t+1}{2^{rt}(rt+1)} \binom{n-t(r-2)}{2t}^{-1} & \text{if } n > rt. \end{cases}$$

Moreover, the algorithm can be implemented to run in polynomial time.

Proof. We note that sets U_i can be computed in polynomial time, and the algorithm recomputes them at most n times. Random contraction can also be implemented in polynomial time, and therefore the overall run-time of the algorithm is polynomial.

We now bound the correctness probability by induction on n. For the base case, we consider $n \leq rt$. In this case, the algorithm returns $\delta(X)$ for a random $X \subseteq V$. There are 2^n possible values for X, and $F = \delta(X)$ for at least one of them. Thus, the probability that the algorithm returns F is at least $\frac{1}{2^n} \geq \frac{1}{2^{rt}} = Q_n$.

Next, we prove the induction step. Suppose $n > rt$. We will need the following claim whose proof appears in the appendix.

▷ Claim 2.2. The algorithm outputs F with probability at least the optimum value of the following linear program.

$$\begin{aligned} \underset{x_2,\ldots,x_r,y_2,\ldots,y_r}{\text{minimize}} \quad & \sum_{j=2}^{r}(x_j - y_j)Q_{n-j+1} && (LP_2)\\ \text{subject to} \quad & 0 \le y_j \le x_j \ \forall j \in \{2,\ldots,r\} \\ & \sum_{j=2}^{r} x_j = 1 \\ & |U_i| \sum_{j=2}^{r} y_j \le \sum_{j=2}^{r} j \cdot x_j \end{aligned}$$

Let U_i be a the largest among the sets $U_1, \ldots, U_t$ that the algorithm generates when executed on input (G, r, t, c, b). Claim 2.2 tells us that the algorithm outputs F with probability at least the optimum value of the linear program (LP_2) from the claim.

The linear program (LP_2) is exactly the linear program (LP_1) from Lemma 1.4 with $\gamma = |U_i|$ and $f(n) := Q_n$. To apply Lemma 1.4, we just need to show that $n \ge |U_i| \ge r + 1$. We recall that U_i is the largest of the t sets that the algorithm constructs. Each of these sets is a subset of V, so we can conclude that $|U_i| \le |V| = n$. We also know from the the construction of the sets $U_1, \ldots, U_t$ that they partition V. This means that $\sum_{j=1}^{t} |U_j| = n$. Since U_i is the largest of the sets, we must have $|U_i| \ge \frac{n}{t}$. Since $n > rt$, this means $|U_i| \ge \frac{rt+1}{t} > r$. Thus $|U_i| > r$, and since r and $|U_i|$ are integers, we conclude that $|U_i| \ge r + 1$. Therefore, we can apply Lemma 1.4 with $\gamma = |U_i|$ to conclude that

$$\Pr[\text{Algorithm returns the cut } F] \ge \min_{2 \le j \le r} \left(1 - \frac{j}{|U_i| - r + j}\right) Q_{n-j+1}.$$

The following claim completes the proof of the theorem. Its proof appears in the appendix. ◀

▷ Claim 2.3. For every $j \in \{2, \ldots, r\}$, we have $\left(1 - \frac{j}{|U_i|-r+j}\right) Q_{n-j+1} \ge Q_n$.

2.2 Finding Multiobjective Min-Cuts

In this section, we present Algorithm 2, which does *not* take a budget-vector as input, yet outputs any multiobjective min-cut (for any choice of budget-vector) with inverse polynomial probability. This is accomplished by returning a collection of cuts.

In contrast to Algorithm 1, all of the randomness in Algorithm 2 (except for the selection of a random cut in the base case) occurs upfront through the selection of a permutation of the hyperedges. Theorem 2.4 summarizes the guarantees of Algorithm 2.

▶ **Theorem 2.4.** *Let G be an r-rank, n-vertex hypergraph with t hyperedge-cost functions. Then, a fixed multiobjective min-cut F is in the collection returned by Algorithm 2 with probability $\frac{\Omega(n^{-2t})}{r2^{rt}}$. Moreover, the algorithm outputs at most n^{t-1} cuts.*

Proof. We begin by showing the second part of the theorem. The algorithm outputs at most one cut for each choice of $n_1, \ldots, n_{t-1} \in [n]$ (or just one cut if $|V| \le rt$). Hence, it outputs at most n^{t-1} cuts.

We now argue that the algorithm retains the same success probability as Algorithm 1, for any fixed budget-vector b.

Algorithm 2 Multiobjective Min-Cut.

Multiobjective-Min-Cut$(G, r, t, c_1, \ldots, c_t)$:

Input: An r-rank hypergraph $G = (V, E)$ and hyperedge-cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$.

If $|V| \leq rt$:
  Pick a random subset X of V and return $\delta(X)$
For $i = 1, \ldots, t$:
  $E_i \leftarrow \{e \in E \colon c_i(e) > 0\}$
  $\pi_i \leftarrow$ a permutation of E_i generated by repeatedly choosing a not yet chosen hyperedge e with probability proportional to $c_i(e)$
$R \leftarrow \emptyset$
For each sequence $n_1, \ldots n_t$ with $n \geq n_1 \geq n_2 \geq \cdots \geq n_{t-1} \geq n_t = rt$:
  $G' \leftarrow G$
  For $i = 1, \ldots, t$:
    While $|V(G')| > n_i$:
      $e \leftarrow$ the first hyperedge from π_i that is still present in G'
      $G' \leftarrow G'/e$
  $X \leftarrow$ a random subset of $V(G')$
  Add $\delta(X)$ to R if it is not already present
Return R

Suppose $n \leq rt$. Then both Algorithm 2 and Algorithm 1 return $\delta(X)$ for a random subset X of the vertices of G. Thus, for any cut F, the two algorithms have the same probability of returning F. Henceforth, we assume $n > rt$.

We will view Algorithm 1 from a different perspective. In that algorithm, whenever we contract a hyperedge e, we choose, for some $i \in [t]$, a hyperedge according to the probability distribution $\Pr[e = e'] = \frac{c_i(e')}{c_i(E)}$. In particular, the choice of i depends on which contractions have been made so far, but the choice of a particular hyperedge, given the choice of i, does not depend on our previous contractions, except for the fact that we do not contract hyperedges which have already been reduced to singletons. We note that allowing the contraction of singletons would not change the success probability of the algorithm.

Therefore, we could modify Algorithm 1 so that it begins by selecting permutations $\pi_1, \ldots, \pi_t$ of $E_1, \ldots, E_t$ (where $E_i = \{e \in E \colon c_i(e) > 0\}$) as in Algorithm 2, and then whenever the algorithm asks to contract a random hyperedge with probability proportional to its weight under c_i, we instead contract the next hyperedge from π_i which is still present in the current hypergraph. This modification does not change at any step the probability that a particular hyperedge is the next contraction of a non-singleton hyperedge, and therefore the success probability of the algorithm remains exactly the same.

Viewing Algorithm 1 in this way, we note that when we reach the base case of $n \leq rt$, we will have contracted the first m_i hyperedges of each π_i, for some $m_1, \ldots, m_t \in \{0, \ldots |E|\}$. The crucial observation now is that interleaved contractions can be separated. That is, if we know m_i for every $i \in [t]$, the order in which we do the contractions is irrelevant: we will get the same resulting hypergraph if we contract the first m_1 hyperedges from π_1, then contract the first m_2 hyperedges from π_2, and so on up through the first m_t hyperedges from π_t instead of the interleaved contractions. Let n_1 be the number of vertices in the hypergraph obtained after contracting the first m_1 hyperedges from π_1, subsequently, let n_2 be the number of vertices in the hypergraph obtained after contracting the first m_2 hyperedges from π_2 and so on.

When we view Algorithm 1 in this way, it is only the choice of the values $n_1, \ldots, n_t$ that depends on the budgets, while the choice of the permutation π_i does not depend on the budgets. Algorithm 2 is running exactly the version of Algorithm 1 that we have just described, except that instead of choosing $n_1, \ldots, n_t$ based on the budgets, it simply tries all possible options (which will certainly include whichever $n_1, \ldots, n_t$ Algorithm 1 would use for the given input budget-vector).

Therefore for every fixed choice of budget-vector b, and every fixed b-multiobjective min-cut F, the probability that F is in the collection R output by the algorithm is at least as large as the probability that F is the cut output by Algorithm 1. By Theorem 2.1, this probability is $\frac{\Omega(n^{-2t})}{r2^{rt}}$, as desired. ◀

We derive Theorem 1.1 from Theorem 2.4 now.

▶ **Theorem 1.1.** *The number of multiobjective min-cuts in an r-rank, n-vertex hypergraph G with t hyperedge-cost functions is $O(r2^{rt}n^{3t-1})$.*

Proof. Let x be the number of multiobjective min-cuts in G. By Theorem 2.4, the expected number of multiobjective min-cuts output by our algorithm Multiobjective Min-Cut (i.e., Algorithm 2) is $(x/r2^{rt})\Omega(n^{-2t})$. Theorem 2.4 also tells us that the algorithm outputs at most n^{t-1} cuts. Therefore, $x = r2^{rt} \cdot O(n^{3t-1})$. ◀

3 Size-Constrained Min-k-Cut in Arbitrary-Rank Hypergraphs

In this section, we consider the problem of finding a minimum cost k-cut subject to constant lower bounds on the weights of the partition classes and prove Theorem 1.3. Throughout this section, we assume that k is a constant. We focus on the cardinality case (i.e., unit-cost variant) for the sake of simplicity of exposition and mention that our algorithm also extends to arbitrary non-negative hyperedge costs.

We begin by formally defining the terminology. Let $G = (V, E)$ be a hypergraph. For a partition $X = (X_1, \ldots, X_k)$ of V, we define $\delta(X)$ to be the set of hyperedges that intersect at least two parts of X. For a weight function $w : V \to \mathbb{Z}_+$, we call (G, w) a *vertex-weighted hypergraph*. We now define our main object of study, namely size-constrained minimum cuts.

▶ **Definition 3.1.** *Let $G = (V, E)$ be a hypergraph, $w : V \to \mathbb{Z}_+$ be a vertex-weight function, $k \geq 2$ be an integer, and $s \in \mathbb{Z}_+^k$ be a size-vector. A k-partition X of V is an* s-size-constrained k-partition *if $w(X_i) \geq s_i$ for every $i \in [k]$. A set of hyperedges $F \subseteq E$ is an* s-size-constrained k-cut *if $F = \delta(X)$ for some s-size-constrained k-partition X. An s-size-constrained k-cut of minimum cardinality is said to be an* s-size-constrained min-k-cut.

The following is the central problem of interest in this section.

s-Size-Constrained Min-k-Cut
Given: A vertex-weighted hypergraph (G, w), a positive integer k, and a size-vector $s \in \mathbb{Z}_+^k$.
Goal: An s-size-constrained min-k-cut.

We give a random contraction based algorithm for this problem. Given a hypergraph $G = (V, E)$ and a size-constraint vector $s \in \mathbb{Z}_+^k$, let $n = |V|$. We define $\sigma_j := \sum_{i=1}^{j} s_i$, and $\alpha_e := \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}$. With these definitions, we solve s-size-constrained min-k-cut using Algorithm 3. We prove Theorem 1.3 using this algorithm.

Algorithm 3 s-Size-Constrained Min-k-Cut.

s-Size-Constrained-Min-k-Cut(G, k, s):

Input: An n-vertex hypergraph $G = (V, E)$, an integer $k \geq 2$ and size-constraint vector $s = (s_1, \ldots, s_k) \in \mathbb{Z}_+$

If $n \leq \max\{2\sigma_{k-1}, \sigma_k\}$
  Pick a random k-partition X of V and return $\delta(X)$
$S \leftarrow$ a random subset of V of size $2\sigma_{k-1}$
$X_i \leftarrow \emptyset$ for $i \in \{1, \ldots, k\}$
For each $v \in S$:
  Pick a random integer $i \in \{1, \ldots, k\}$ and add v to X_i
$X_k \leftarrow X_k \cup (V \setminus S)$
$X \leftarrow (X_1, \ldots, X_k)$
If X is a k-partition of V:
  $R \leftarrow \delta(X)$
Else:
  $R \leftarrow E$
Compute $\alpha_e := \frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}}$ for every $e \in E$
If $\alpha_e = 0$ for every hyperedge $e \in E(G)$:
  Return R
$e \leftarrow$ a random hyperedge of G chosen with probability proportional to α_e
$R' \leftarrow$ s-Size-Constrained-Min-k-Cut$(G/e, k, s)$
Return R with probability $\frac{1}{n}$ and R' with probability $\frac{n-1}{n}$

▶ **Theorem 1.3.** *Let $k \geq 2$ be a fixed positive integer and let $1 \leq s_1 \leq s_2 \leq \ldots \leq s_k$ be fixed positive integers. Let G be an n-vertex hypergraph with hyperedge-cost function $c : E \to \mathbb{R}_+$. Then, there exists a polynomial-time algorithm that takes (G, c) as input and returns a fixed w-weighted s-size-constrained min-k-cut for any choice of vertex-weight function $w : V \to \mathbb{Z}_+$ with probability $\Omega\left(\frac{1}{n^{2\sigma_{k-1}+1}}\right)$, where $\sigma_{k-1} := \sum_{i=1}^{k-1} s_i$.*

Proof. We consider Algorithm 3. We first analyze its run-time. Each recursive call reduces the number of vertices in the hypergraph. Thus, the algorithm makes at most n recursive calls. Apart from the recursion steps, the algorithm only selects random partitions and performs contractions, both of which can be implemented to run in polynomial time.

Now we analyze the success probability. Let $\mathcal{G}_n$ be the set of all vertex-weighted n-vertex hypergraphs which contain an s-size-constrained k-cut. For a vertex-weighted hypergraph (G, w), let $M(G, w)$ be the set of all s-size-constrained min-k-cuts in (G, w). Define

$$q_n := \min_{(G,w)\in\mathcal{G}_n} \min_{F\in M(G,w)} \Pr[\text{Algorithm returns } F \text{ on input } (G, w, k, s)], \text{ and}$$

$$Q_n := \begin{cases} \left(k^{\max\{2\sigma_{k-1},\sigma_k\}}\right)^{-1} & \text{if } n \leq \max\{2\sigma_{k-1}, \sigma_k\}, \text{and} \\ \left(k^{\max\{2\sigma_{k-1},\sigma_k\}} n\binom{n}{2\sigma_{k-1}}\right)^{-1} & \text{if } n > \max\{2\sigma_{k-1}, \sigma_k\}. \end{cases}$$

We note that $Q_n = \Omega(k^{-\sigma_{k-1}-\sigma_k} n^{-2\sigma_{k-1}-1})$, so it suffices to show that $q_n \geq Q_n$ for all $n \geq k$ (for smaller n, there are no k-cuts).

We proceed by induction on n. Let F be an s-size-constrained min-k-cut in (G, w). Let Y be an s-size-constrained k-partition with $F = \delta(Y)$. We assume that $|Y_1| \leq |Y_2| \leq \cdots \leq |Y_k|$. This assumption is without loss of generality because we can relabel the parts of an s-size-constrained k-partition so that they are in increasing order of size and the resulting partition will still be an s-size-constrained k-partition (since we have assumed that the size vector s is in increasing order).

For the base case, suppose $n \leq \max\{2\sigma_{k-1}, \sigma_k\}$. In this case, the algorithm returns $\delta(X)$ where X is a k-partition of V chosen uniformly at random. Since $F = \delta(Y)$, the probability that $F = \delta(X)$ for the X randomly chosen by the algorithm is at least $\Pr[X = Y]$. The number of k-partitions of V is at most k^n, the number of ways to assign each of the n vertices to one of k labeled sets. Thus, $\Pr[X = Y] \geq k^{-n} \geq k^{-max(2\sigma_{k-1}, \sigma_k)} = Q_n$.

Now we will prove the inductive step. Assume that $n > \max\{2\sigma_{k-1}, \sigma_k\}$. By the inductive hypothesis, we have $q_{n'} \geq Q_{n'}$ for all $n' \in \{k, \ldots, n-1\}$. We will show that $q_n \geq Q_n$.

Suppose $|Y_k| \geq n - 2\sigma_{k-1}$. Let T be an arbitrary subset of Y_k of size $n - 2\sigma_{k-1}$. Consider the set S chosen by the algorithm. The probability that S is equal to $V - T$ is $\binom{n}{2\sigma_{k-1}}^{-1}$. Next, consider the sets X_i created by the algorithm. The probability that $X_i = Y_i$ for every $i \in [k]$ conditioned on $S = V - T$, is $k^{-2\sigma_{k-1}}$. Thus, the probability that the k-partition X obtained in the algorithm is identical to Y is at least $k^{-2\sigma_{k-1}}\binom{n}{2\sigma_{k-1}}^{-1}$. Since the last step of the algorithm returns $R = \delta(X)$ with probability $1/n$, it follows that the algorithm returns F with probability at least $\left(nk^{\max\{2\sigma_{k-1}, \sigma_k\}}\binom{n}{2\sigma_{k-1}}\right)^{-1} = Q_n$.

Henceforth, we assume that $|Y_k| < n - 2\sigma_{k-1}$. We will call a hyperedge *large* if it contains at least $n - 2\sigma_{k-1}$ vertices. Since Y_k is the largest part of the k-partition Y, every large hyperedge must be contained in the k-cut F. In particular, if $\alpha_e = 0$ for a hyperedge e, then $\binom{n-|e|}{\sigma_{k-1}} = 0$, which implies that $n - |e| < \sigma_{k-1}$, and hence e is large and consequently, e cannot be in F.

Next, suppose that $\alpha_e = 0$ for every hyperedge e. Then every hyperedge is a large hyperedge and therefore, $F = E$. In this case, the algorithm will return R. We note that $R = E$ if X is not a k-partition. We lower bound the probability that X is not a k-partition now. If all vertices in S are assigned to X_k, then X is not a k-partition. The probability that all vertices in S are assigned to X_k is $k^{-2\sigma_{k-1}}$. Thus, the probability that the algorithm returns $F = R = E$ is at least $k^{-2\sigma_{k-1}} \geq Q_n$.

Henceforth, we assume that $\alpha_e > 0$ for some hyperedge $e \in E$. This means that the algorithm will contract some hyperedge and then recurse on the resulting hypergraph. Let e' be a random variable for the hyperedge chosen to be contracted. Let w' be the weight function defined on the vertices of G/e' as follows: $w'(v) := w(v)$ for each $v \in V \setminus e'$ and $w'(v) := \sum_{u \in e'} w(u)$ when v is the new vertex resulting from the contraction. If $e' \notin F$, then F will be an s-size-constrained min-k-cut in $(G/e', w')$. Therefore, we have that $\Pr[R' = F \text{ on input } (G, k, s)]$

$$
\begin{aligned}
&= \sum_{e \in E \setminus F} \Pr[e' = e] \cdot \Pr[\text{Algorithm returns } F \text{ on input } (G/e, k, s)] \\
&\geq \sum_{e \in E \setminus F} \frac{\alpha_e}{\sum_{f \in E} \alpha_f} \cdot q_{n-|e|+1}.
\end{aligned}
$$

Let e be a hyperedge that is not in the k-cut F. Then, e cannot be a large hyperedge and hence, $|e| < n - 2\sigma_{k-1}$. Consequently, $n - |e| + 1 > 2\sigma_{k-1} + 1 \geq k$. Therefore, by applying the induction hypothesis, we have $q_{n-|e|+1} \geq Q_{n-|e|+1}$. Hence,

$$\Pr[R' = F \text{ on input } (G,k,s)] \geq \frac{1}{\sum_{f\in E}\alpha_f}\sum_{e\in E\setminus F}\alpha_e \cdot Q_{n-|e|+1}.$$

We need the following two claims. We prove Claim 3.2 in the full version of this paper, and we defer the proof of Claim 3.3 to complete the proof of the theorem.

▷ Claim 3.2. For every hyperedge $e \in E \setminus F$, we have $\alpha_e Q_{n-|e|+1} \geq \frac{nQ_n}{n-1}$.

▷ Claim 3.3. $\frac{|E\setminus F|}{\sum_{f\in E}\alpha_f} \geq 1$.

By Claim 3.2, we have that $\Pr[R' = F \text{ on input } (G,k,s)]$

$$= \frac{1}{\sum_{f\in E}\alpha_f}\sum_{e\in E\setminus F}\alpha_e \cdot Q_{n-|e|+1} \geq \frac{1}{\sum_{f\in E}\alpha_f}\sum_{e\in E\setminus F}\frac{nQ_n}{n-1} = \frac{|E\setminus F|}{\sum_{f\in E}\alpha_f}\cdot\frac{nQ_n}{n-1}.$$

Thus, Claim 3.3 implies that

$$\Pr[R' = F \text{ on input } (G,k,s)] \geq \frac{nQ_n}{n-1}.$$

Finally, we note that since we have assumed $n > \max\{2\sigma_{k-1}, \sigma_k\}$ and $\alpha_e > 0$ for some e, the probability that the algorithm returns R' is $(n-1)/n$. Thus, we conclude that $\Pr[\text{Algorithm returns } F] \geq Q_n$. ◀

Proof of Claim 3.3. Let $Z = (Z_1, \ldots, Z_k)$ be a random k-partition obtained by picking disjoint sets $Z_1, \ldots, Z_{k-1}$ with $|Z_i| = s_i$ and setting $Z_k = V \setminus \bigcup_{i=1}^{k-1} Z_i$. Since $n > \sigma_k$ and every vertex has weight at least 1, the k-partition Z is an s-size-constrained k-partition. Therefore, $|\delta(Z)|$ is an upper bound on $|F|$. In particular, $|F| \leq \mathbb{E}(|\delta(Z)|) = \sum_{e\in E}\Pr(e \in \delta(Z))$. Negating the inequality and adding $|E|$ to both sides gives

$$\begin{aligned}|E\setminus F| &\geq \sum_{e\in E}(1-\Pr(e\in\delta(Z))) = \sum_{e\in E}\Pr(e\notin\delta(Z)) = \sum_{e\in E}\sum_{i=1}^{k}\Pr(e\subseteq Z_i)\\ &\geq \sum_{e\in E}\frac{\binom{n-|e|}{\sigma_{k-1}}}{\binom{n}{\sigma_{k-1}}} = \sum_{e\in E}\alpha_e.\end{aligned}$$

◁

▶ Remark. Since our algorithm does not even take the vertex-weights as input, it could trivially be extended to handle a version of the problem where we have multiple weight functions on the vertices (as in the previous sections) each with their own minimum sizes. If we have t vertex-weight functions, $w_1, \ldots, w_t : V \to \mathbb{Z}_+$ and each function w_j has an associated list of lower bounds $s_{j,1}, \ldots, s_{j,k}$, then we can find a min-k-cut satisfying all of these lower-bound constraints with at least inverse polynomial probability by simply running our algorithm with $s_i = \max_{j\in[t]} s_{j,i}$ for every $i \in [t]$.

4 Conclusion and Open problems

In this work, we illustrated the versatility of the random contraction technique by addressing multicriteria versions of min-cut and size-constrained min-k-cut problems. There are several interesting open questions in this area. We conclude by stating a few: (1) For the number of pareto-optimal cuts and multiobjective min-cuts, there is still a gap between our lower

bound (which is $\Omega(n^t)$) and our upper bound (which is $O(n^{3t-1})$). Can we improve either of these bounds? We believe that improving our bounds for the number of b-multiobjective min-cuts for a fixed budget-vector $b \in \mathbb{R}_+^{t-1}$ would be a first-step towards this goal. (2) We gave a polynomial-time algorithm to solve the b-multiobjective min-cut problem in constant-rank hypergraphs. How about arbitrary-rank hypergraphs? Is the b-multiobjective min-cut problem in arbitrary rank hypergraphs (even for $t = 2$ criteria) solvable in polynomial-time or is it NP-hard?

References

1 H. Aissi, A. Mahjoub, T. McCormick, and M. Queyranne. Strongly polynomial bounds for multiobjective and parametric global minimum cuts in graphs and hypergraphs. *Mathematical Programming (Preliminary version in IPCO 2014)*, 154(1-2):3–28, 2015.

2 H. Aissi, A. Mahjoub, and R. Ravi. Randomized Contractions for Multiobjective Minimum Cuts. In *Proceedings of the 25th Annual European Symposium on Algorithms*, ESA, pages 6:1–6:13, 2017.

3 A. Armon and U. Zwick. Multicriteria global minimum cuts. *Algorithmica*, 46(1):15–26, 2006.

4 C. Beideman, K. Chandrasekaran, and C. Xu. Multicriteria Cuts and Size-Constraiend k-Cuts. arXiv, 2020. `arXiv:2006.11589`.

5 K. Chandrasekaran, C. Xu, and X. Yu. Hypergraph k-cut in randomized polynomial time. *Mathematical Programming (Preliminary version in SODA 2018)*, November 2019.

6 C. Chekuri and C. Xu. Minimum cuts and sparsification in hypergraphs. *SIAM Journal on Computing*, 47(6):2118–2156, 2018.

7 E. Dinits, A. Karzanov, and M. Lomonosov. On the structure of a family of minimal weighted cuts in a graph. *Studies in Discrete Optimizatoin (in Russian)*, pages 290–306, 1976.

8 M. Ghaffari, D. Karger, and D. Panigrahi. Random contractions and sampling for hypergraph and hedge connectivity. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, pages 1101—-1114, 2017.

9 M. Goemans and J. Soto. Algorithms for symmetric submodular function minimization under hereditary constraints and generalizations. *SIAM Journal on Discrete Mathematics*, 27(2):1123–1145, 2013.

10 O. Goldschmidt and D. Hochbaum. A Polynomial Algorithm for the k-cut Problem for Fixed k. *Mathematics of Operations Research*, 19(1):24–37, February 1994.

11 F. Guinez and M. Queyranne. The size-constrained submodular k-partition problem. Unpublished manuscript. Available at `https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxmbGF2aW9ndWluZXpob21lcGFnZXxneDo0NDVlMThkMDg4ZWRlOGI1`. See also `https://smartech.gatech.edu/bitstream/handle/1853/43309/Queyranne.pdf`, 2012.

12 D. Karger. Global Min-Cuts in RNC, and Other Ramifications of a Simple Min-Cut Algorithm. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, page 21–30, 1993.

13 D. Karger. Enumerating parametric global minimum cuts by random interleaving. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing*, STOC, pages 542–555, 2016.

14 D. Kogan and R. Krauthgamer. Sketching cuts in graphs and hypergraphs. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, ITCS, pages 367–376, 2015.

15 K. Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28(4):1460–1509, 1999.

A Appendix

A.1 Proof of Claim 2.2

Proof of Claim 2.2. Since $n > rt$, when the algorithm is executed on G it will contract a randomly chosen hyperedge and recurse. Let e' be the random hyperedge chosen by the algorithm. If $e' \notin F$, then F will still be a b-multiobjective min-cut in G/e'. We observe that G/e' is a hypergraph with $n - |e'| + 1$ vertices and the rank of G/e' is at most the rank of G. Therefore, if $e' \notin F$, then the algorithm will output F with probability at least $Q_{n-|e'|+1}$.

Let $i \in [t]$ be the index of the cost function chosen by the algorithm. Let

$$E_j := \{e \in E \colon |e| = j\},$$
$$x_j := \Pr[e' \in E_j] = \frac{c_i(E_j)}{c_i(E)}, \text{ and}$$
$$y_j := \Pr[e' \in E_j \cap F] = \frac{c_i(E_j \cap F)}{c_i(E)}.$$

We note that E_j is the set of hyperedges of size j, x_j is the probability of picking a hyperedge of size j to contract, and y_j is the probability of picking a hyperedge of size j from F to contract. We know that

$$\Pr[\text{Algorithm returns the cut } F] \geq \sum_{j=2}^{r} (x_j - y_j) Q_{n-j+1}. \tag{5}$$

The values of x_j and y_j will depend on the structure of G. However we can deduce some relationships between them. Since $0 \leq c_i(E_j \cap F) \leq c_i(E_j)$ for every $j \in \{2, \ldots, r\}$, we know that

$$0 \leq y_j \leq x_j \text{ for every } j \in \{2, \ldots, r\}. \tag{6}$$

Moreover, x_j is the probability of picking a hyperedge of size j. Hence,

$$\sum_{j=2}^{r} x_j = 1. \tag{7}$$

Next, we show that for every $i \in [t]$ and every $v \in U_i$, we have

$$c_i(F) \leq c_i(\delta(v)). \tag{8}$$

If $i < t$, then $c_i(F) \leq b_i < c_i(\delta(v))$ for every $v \in U_i$. Let $i = t$. We recall that F is a b-multiobjective min-cut. Since every cut induced by a single vertex in U_t satisfies all of the budgets, no such cut can have a better c_t-cost than F, so again $c_i(F) \leq c_i(\delta(v))$ for every $v \in U_i$.

From inequality (8), we conclude that

$$c_i(F) \leq \frac{\sum_{v \in U_i} c_i(\delta(v))}{|U_i|} \leq \frac{\sum_{v \in V} c_i(\delta(v))}{|U_i|} = \frac{\sum_{e \in E} |e| c_i(e)}{|U_i|} = \frac{\sum_{j=2}^{r} j \cdot c_i(E_j)}{|U_i|}.$$

Therefore

$$\sum_{j=2}^{r} y_j = \Pr[e' \in F] = \frac{c_i(F)}{c_i(E)} \leq \frac{1}{|U_i|} \sum_{j=2}^{r} j \cdot x_j.$$

Thus, we have that

$$|U_i| \sum_{j=2}^{r} y_j \leq \sum_{j=2}^{r} j \cdot x_j. \tag{9}$$

The minimum value of our lower bound in equation (5) over all choices of x_j and y_j that satisfy inequalities (6), (7), and (9) is a lower bound on the probability that the algorithm outputs F. ◁

A.2 Proof of Claim 2.3

Proof of Claim 2.3. Let $j \in \{2, \ldots, r\}$. The given inequality is equivalent to $\frac{Q_{n-j+1}}{Q_n} \geq \frac{|U_i|-r+j}{|U_i|-r}$. Since U_i is the largest among $U_1, \ldots, U_t$ which together partition V, we have $|U_i| \geq \frac{n}{t}$. Consequently, $\frac{|U_i|-r+j}{|U_i|-r} = 1 + \frac{j}{|U_i|-r} \leq 1 + \frac{jt}{n-rt}$. Therefore, it suffices to prove that $\frac{Q_{n-j+1}}{Q_n} \geq 1 + \frac{jt}{n-tr}$. We case on the value of $n-j+1$.

1. Suppose that $n - j + 1 > rt$. Then, we have

$$\frac{Q_{n-j+1}}{Q_n} = \frac{\binom{n-t(r-2)}{2t}}{\binom{n-j+1-t(r-2)}{2t}} = \prod_{\ell=0}^{2t-1} \frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell}. \tag{10}$$

We consider two sub-cases based on the value of j.

a. Suppose that $j > 2t$. Then, we observe that

$$\begin{aligned}
\prod_{\ell=0}^{2t-1} \frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell} &\geq \left(\frac{n-t(r-2)}{n-j+1-t(r-2)}\right)^{2t} \\
&= \left(1 + \frac{j-1}{n-j+1-t(r-2)}\right)^{2t} \\
&\geq 1 + \frac{2t(j-1)}{n-j+1-t(r-2)} \\
&\geq 1 + \frac{jt+(j-2)t}{n-rt} \\
&\geq 1 + \frac{jt}{n-rt}.
\end{aligned}$$

We use $j > 2t$ in the second to last inequality and $j \geq 2$ in the final inequality.

b. Suppose that $j \leq 2t$. Then we can cancel additional terms from the the right hand side of equation (10) to obtain that

$$\begin{aligned}
\prod_{\ell=0}^{2t-1} \frac{n-t(r-2)-\ell}{n-j+1-t(r-2)-\ell} &= \prod_{\ell=0}^{j-2} \frac{n-t(r-2)-\ell}{n-tr-\ell} \\
&\geq \left(\frac{n-t(r-2)}{n-tr}\right)^{j-1} \\
&= \left(1 + \frac{2t}{n-rt}\right)^{j-1} \\
&\geq 1 + \frac{2t(j-1)}{n-rt} \\
&\geq 1 + \frac{jt}{n-rt}.
\end{aligned}$$

Thus, in either subcase, our desired inequality holds.

2. Suppose that $n - j + 1 \leq rt$. Now the expression for Q_{n-j+1} is different. Since we still know that $n \geq rt + 1$ and $j \leq r$, we conclude that

$$\begin{aligned}\frac{Q_{n-j+1}}{Q_n} &= \frac{(rt+1)\binom{n-t(r-2)}{2t}}{2t+1} \\ &\geq \frac{(rt+1)\binom{rt+1-t(r-2)}{2t}}{2t+1} \\ &= rt+1 \\ &\geq 1+jt \\ &\geq 1+\frac{jt}{n-rt}.\end{aligned}$$

Thus, our desired inequality holds in all cases. ◁

A.3 Enumerating Multiobjective Min-Cuts and Pareto-Optimal Cuts

In this section, we give algorithms to enumerate all multiobjective min-cuts and pareto-optimal cuts in polynomial time.

We first give a polynomial time algorithm to enumerate all multiobjective min-cuts. We execute our algorithm for Multiobjective Min-Cut (i.e., Algorithm 2) a sufficiently large number of times so that it succeeds with high probability (i.e., with probability at least $1 - 1/n$): In particular, executing it $r^2 t 2^{rt} O(n^{2t} \log n)$ many times gives us a collection $\mathcal{C}$ that is a superset of the collection $\mathcal{C}_{MO}$ of multiobjective min-cuts with high probability. Moreover, the size of the collection $\mathcal{C}$ is $r^2 t 2^{rt} O(n^{3t-1} \log n)$. We can prune $\mathcal{C}$ to identify $\mathcal{C}_{MO}$ in polynomial-time as follows: remove every cut $F \in \mathcal{C}$ for which there exists a cut $F' \in \mathcal{C}$ with $c_t(F') < c_t(F)$ and $c_i(F') \leq c_i(F)$ for every $1 \leq i \leq t-1$.

Next, we give a polynomial time algorithm to enumerate pareto-optimal cuts. By containment relation (1), it suffices to identify all pareto-optimal cuts in the collection $\mathcal{C}$. For this, we only need a polynomial-time procedure to verify if a given cut F is pareto-optimal. Algorithm 4 gives such a procedure. It essentially searches for a cut that dominates the given cut F by running our algorithm for b-Multiobjective Min-Cut with t different budget-vectors.

Algorithm 4 Verify pareto-optimality of a given cut.

Verify-Pareto-Optimality$(G, r, t, c_1, \ldots, c_t, F)$:
Input: An r-rank hypergraph $G = (V, E)$, cost functions $c_1, \ldots, c_t : E \to \mathbb{R}_+$, and a cut F in G

For $i = 1, \ldots, t$:
  $\vec{c} \leftarrow (c_1, \ldots c_{i-1}, c_{i+1}, \ldots, c_t, c_i)$
  $\vec{b} \leftarrow (c_1(F), \ldots, c_{i-1}(F), c_{i+1}(F), \ldots, c_t(F))$
  For $j = 1, \ldots, r2^{rt}O(n^{2t}\log(n))$:
    $F' \leftarrow b$-Multiobjective-Min-Cut$(G, r, t, \vec{c}, \vec{b})$
    If F' is a b-multiobjective cut in $(G, \vec{c})$ and $c_i(F') < c_i(F)$:
      Return FALSE
Return TRUE

▶ **Theorem A.1.** *Given an r-rank, n-vertex hypergraph G with t hyperedge-cost functions and a cut F in G, if F is a pareto-optimal cut, then Algorithm 4 returns TRUE, and if F is*

not a pareto-optimal cut, then the algorithm returns FALSE with high probability. Moreover, the algorithm can be implemented to run in polynomial time.

Proof. The run-time of the algorithm is polynomial because our algorithm for b-Multiobjective Min-Cut (i.e., Algorithm 1) is a polynomial-time algorithm. Algorithm 4 returns false only if it finds a cut that dominates the input cut F. If F is pareto-optimal, no such cut will exist, and therefore the algorithm will return true.

Next, suppose the input cut F is not pareto-optimal. Then F must be dominated by some other cut F'. Let $i \in [t]$ be such that $c_i(F') < c_i(F)$ (such an i is guaranteed to exist by the definition of domination). Let b be a budget-vector of the costs of F under the cost functions other than c_i and let c' be c with c_i moved to the end of the vector of cost functions. Then F will not be a b-multiobjective min-cut in (G, c'), since F' also satisfies b, but has a lower c_i-cost.

Therefore, any b-multiobjective min-cut will dominate F (since it will also satisfy b and cannot have higher c_i cost than F'). Thus, if any of our $r2^{rt}n^{2t}\log(n)$ calls to our algorithm for b-Multiobjective-Min-Cut (i.e., Algorithm 1) for this value of i returns a multiobjective min-cut, then the algorithm will return false. By Theorem 2.1, our algorithm for b-Multiobjective Min-Cut returns a b-multiobjective min-cut with probability $\frac{1}{r2^{rt}}\Omega(\frac{1}{n^{2t}})$. Therefore, if we run this algorithm $r2^{rt}O(n^{2t}\log(n))$ times, a b-multiobjective min-cut will be returned at least once with high probability, and our algorithm will correctly return false. ◀

A.4 Lower Bounds

In this section we discuss lower bounds on the number of distinct pareto-optimal cuts in n-vertex hypergraphs. Karger gave a family of graphs with $n^{t/2}$ parametric min-cuts [13]. We recall that every parametric min-cut is a pareto-optimal cut by the containment relation (1). Thus, $n^{t/2}$ is also a lower bound on the number of pareto-optimal cuts in n-vertex hypergraphs. To the best of the authors' knowledge, this is the best lower bound on the number of pareto-optimal cuts that is known in the literature. We give an $\Omega(n^t)$ (for constant t) lower bound on the number of pareto-optimal cuts in a graph. Our lower bound construction is different from that of Karger.

▶ **Theorem A.2.** *For all positive integers t and n such that $n \geq t+2$, there exists an n-vertex graph G with associated edge-cost functions $c_1, \ldots, c_t : E(G) \to \mathbb{R}_+$ such that G has at least $\left(\frac{n-2}{t}\right)^t$ distinct pareto-optimal cuts.*

Proof. For fixed n and t, construct a graph G as follows. The graph G has two special vertices u and v. The rest of the vertices are used to form t distinct paths between u and v with each path consisting of at least $\lfloor\frac{n-2}{t}\rfloor + 1 > \frac{n-2}{t}$ distinct edges. We assign edge costs as follows: If e is an edge in the i'th path, then $c_i(e) = 1$, while $c_j(e) = 1/(t+1)$ for every $j \in [t] \setminus \{i\}$. See Figure 1 for an example. We will show that any cut which contains exactly one edge from each path is pareto-optimal. The number of such cuts is at least $\left(\frac{n-2}{t}\right)^t$, since each path has at least $(n-2)/t$ edges, so this will suffice to prove the theorem.

We observe that any cut contains either exactly one edge from each path or at least two edges from some path. Any cut F which contains exactly one edge from each path will have $c_i(F) = 2t/(t+1)$ for every $i \in [t]$. Any cut F' that contains at least two edges from some path $i \in [t]$ will have $c_i(F') = 2 > 2t/(t+1)$. Therefore no cut which contains two edges from the same path can dominate a cut which contains exactly one edge from each path. Furthermore, if two different cuts each contain exactly one edge from all paths, then they both have the same cost under every cost function, and thus neither can dominate the other. We conclude that every cut which contains exactly one edge from each path is pareto-optimal. ◀

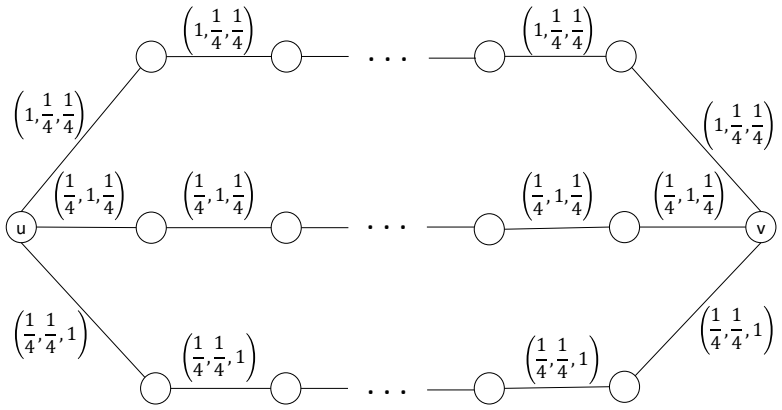


Figure 1 An illustration of our lower bound construction for $t = 3$.

▶ Remark A.3. The lower bound from Theorem A.2 is still significantly smaller than the $O(n^{3t-1})$ upper bound from Theorem 1.1. We believe that this gap comes from the slack in the analysis of our randomized algorithms.

▶ Remark A.4. We note that the construction in Theorem A.2 also shows that there exists a budget-vector $b \in \mathbb{R}_+^{t-1}$ such that the number of b-multiobjective min-cuts is $\Omega(n^t)$: consider budget values $b_i = (2t)/(t+1)$ for every $i \in [t-1]$. We emphasize that since not every multiobjective min-cut is pareto-optimal, this lower bound does not imply the one from Theorem A.2. Since distinct pareto-optimal cuts need not be b-multiobjective min-cuts for the same vector b, the bound in Theorem A.2 does not immediately imply this bound either.

On Testing and Robust Characterizations of Convexity

Eric Blais
University of Waterloo, Canada
eric.blais@uwaterloo.ca

Abhinav Bommireddi
University of Waterloo, Canada
vabommir@uwaterloo.ca

Abstract

A body $K \subset \mathbb{R}^n$ is convex if and only if the line segment between any two points in K is completely contained within K or, equivalently, if and only if the convex hull of a set of points in K is contained within K. We show that neither of those characterizations of convexity are robust: there are bodies in $\mathbb{R}^n$ that are far from convex – in the sense that the volume of the symmetric difference between the set K and any convex set C is a constant fraction of the volume of K – for which a line segment between two randomly chosen points $x, y \in K$ or the convex hull of a random set X of points in K is completely contained within K except with exponentially small probability. These results show that any algorithms for testing convexity based on the natural line segment and convex hull tests have exponential query complexity.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Convexity, Line segment test, Convex hull test, Intersecting cones

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.18

Category RANDOM

Acknowledgements The authors thank Lap Chi Lau for numerous insightful discussions during the course of this research. The authors also thank the anonymous referees for valuable feedback and suggestions.

1 Introduction

A *body* is a subset of $\mathbb{R}^n$ that is compact – i.e., closed and bounded – and has a non-empty interior. A body $K \subset \mathbb{R}^n$ is *convex* if for every two points $x, y \in K$ and every parameter λ in the range $[0, 1]$, the point $z = \lambda x + (1 - \lambda)y$ is also in K. The *geometric convexity testing* problem is a formalization of the following property testing problem:

> How efficiently can we distinguish convex bodies from those that are far from convex?

The geometric convexity testing problem was first studied by Rademacher and Vempala [18], who formalized the problem as follows. A body $K \subset \mathbb{R}^n$ is *ϵ-far from convex* for some $\epsilon > 0$ if for every convex body $C \subset \mathbb{R}^n$, the volume of the symmetric difference of K and C is bounded below by $\mathrm{Vol}(K \triangle C) \geq \epsilon \,\mathrm{Vol}(K)$. Following the standard framework of property testing [20, 12], we can then define an *ϵ-tester for convexity* to be a bounded-error randomized algorithm that distinguishes convex bodies from bodies that are ϵ-far from convex. We consider testers that access an unknown body $K \subset \mathbb{R}^n$ via the following two standard oracles:

Membership oracle. Given as input a point $x \in \mathbb{R}^n$, the oracle returns "yes" if and only if $x \in K$.

Random oracle. The oracle returns a point x drawn uniformly at random from K.


© Eric Blais and Abhinav Bommireddi;
licensed under Creative Commons License CC-BY



The measure of complexity of ϵ-testers of convexity that we examine is the minimum number of queries to either of these oracles that they require, and the main open question is whether there exists an ϵ-tester of convexity of bodies in $\mathbb{R}^n$ that has query complexity that is polynomial in both n and $1/\epsilon$. Three natural testers have been proposed and studied previously with the aim of answering this question.

I. Approximation tester

Rademacher and Vempala [18] showed that there is an ϵ-tester for convexity of bodies in $\mathbb{R}^n$ with query complexity $q = (cn/\epsilon)^n$ for some constant $c > 0$. This tester is obtained via the natural *testing by learning* approach [12]. With q queries to the random oracle for K, we obtain a set S of points whose convex hull C satisfies $\mathrm{Vol}(K \triangle C) < \frac{\epsilon}{2}\mathrm{Vol}(K)$ with high probability when K is convex and (by definition) always satisfies $\mathrm{Vol}(K \triangle C) \geq \epsilon\,\mathrm{Vol}(K)$ when K is ϵ-far from convex. A tester can then distinguish between these two cases with $O(1/\epsilon)$ queries to the random and membership oracles for K.

However, any ϵ-tester for convexity that follows the testing by learning approach must have query complexity exponential in n, since any algorithm that learns a convex set C which satisfies $\mathrm{Vol}(K \triangle C) \leq \epsilon\,\mathrm{Vol}(K)$ for some unknown convex body $K \subset \mathbb{R}^n$ must have query complexity $2^{\Omega(\sqrt{n}/\epsilon)}$ [13]. (In fact, a number of queries that is exponential in n is required even just for estimating the volume of K [2, 11].) So a completely different approach is required if we aim to test convexity of high-dimensional bodies more efficiently.

II. Line segment tester

The definition of convex bodies immediately suggests a simple *line segment test* for convexity: draw two points $x, y \in K$ using the random oracle for K, pick a parameter $\lambda \in [0, 1]$ according to some distribution, and use the membership oracle to determine if the point $z = \lambda x + (1 - \lambda)y$ is in K. If K is convex, this test will always pass, and conversely when K is ϵ-far from convex then there must exist some points x, y, z for which this test does not pass.

A natural idea for constructing an ϵ-tester for convexity is to simply run the line segment test multiple times and accept if and only if each test passes. But Rademacher and Vempala [18] showed that the resulting tester cannot have query complexity that is polynomial in *both* n and $1/\epsilon$. More precisely, they showed that there is a body $K \subset \mathbb{R}^n$ which is $\Omega(\frac{1}{n^2})$-far from convex but for which the line segment $\overline{xy}$ joining two points x and y drawn uniformly at random from K satisfies $\Pr[\overline{xy} \not\subseteq K] = 2^{-\Omega(n)}$.

The counter-example of Rademacher and Vempala, however, does not rule out the possibility that there exists an ϵ-tester of convexity with query complexity polynomial in n when ϵ is a constant. Our first result rules out this possibility as well, showing that the query complexity of testers obtained from the line segment test must be exponential in n for all $\epsilon \leq \frac{1}{8}$.

▶ **Theorem 1.1.** *There exists a body $K \subset \mathbb{R}^n$ that is $\frac{1}{8}$-far from convex for which*

$$\Pr_{x,y \in K}\left[\overline{xy} \not\subseteq K\right] = 2^{-\Omega(n)}. \tag{1}$$

Theorem 1.1 has another interpretation that is independent of property testing: it says that the line segment characterization of convexity is not *robust*: while it is true that only convex bodies satisfy $\overline{xy} \in K$ for every $x, y \in K$, there are bodies that are far from convex where $\overline{xy} \in K$ still holds for "most" points in K.

III. Convex hull tester

For any $m \geq 1$, the *convex hull* of a set $X = \{x^{(1)}, \ldots, x^{(m)}\}$ of m points in $\mathbb{R}^n$ is

$$\mathrm{conv}\Big(X\Big) := \left\{ y : \exists \lambda_1, \ldots, \lambda_m \geq 0, \sum_{i=1}^{m} \lambda_i = 1 \text{ s.t. } y = \sum_{i=1}^{m} \lambda_i x^{(i)} \right\},$$

the set of points that can be obtained by taking a convex combination of the points in X. A natural extension of the line segment test is the *convex hull test*: for some $m \geq 2$, draw a set X of m points from K using the random oracle, draw a point z from $\mathrm{conv}(X)$ according to some distribution, and check whether z is in K. When $m = 2$, the convex hull test is equivalent to the line test which, as we have seen above, cannot lead to an efficient tester for convexity. For $m \geq 3$, however, it is possible that it leads to much more efficient testing algorithms. Indeed, Berman, Murzabulatov, and Raskhodnikova [6, 5] showed that in a slightly different property testing model, the convex hull test can be used to test convexity with a number of queries that is polynomial in $1/\epsilon$ in the two-dimensional setting where $K \subset \mathbb{R}^2$. (See also [19, 7] for related results.)

Our next and main result rules out the possibility of obtaining an efficient tester for convexity in the high-dimensional setting using the convex hull test by showing that the convex hull characterization is not robust, even when taking the convex hull of an *exponential* (in the dimension n) number of points.

▶ **Theorem 1.2.** *There exist a body $K \subset \mathbb{R}^n$ that is $\frac{1}{8}$-far from convex and a constant $c > 0$ such that a set $X = \{x^{(1)}, \ldots, x^{(m)}\} \in K$ of $m = 2^{cn}$ points drawn uniformly and independently at random from K satisfies*

$$\Pr_X \Big[\mathrm{conv}(X) \not\subseteq K\Big] = 2^{-\Omega(n)}.$$

Theorem 1.2 shows that any ϵ-tester for convexity built on the convex hull test must have query complexity $2^{\Omega(n)}$. It also relates to a conjecture of Rademacher and Vempala [18]: they conjecture that when $K \subset \mathbb{R}^n$ is ϵ-far from convex and $x, y, z \in K$ are drawn uniformly at random from K, then the intersection of K with the two-dimensional subspace spanned by x, y, and z is non-convex with probability at least $\Omega(\epsilon/n)$. Theorem 1.2 shows that it is impossible to strengthen the conjecture by replacing the subspace spanned by $\{x, y, z\}$ with the convex hull of these points, since in that case the resulting statement is false.

1.1 Proof overview

Theorems 1.1 and 1.2 are established constructively. The construction that achieves the bounds promised in the theorems is obtained by taking the union of two truncated cones, as pictured in Figure 1. The main technical component of the proof of the theorems lies in the task of showing that, contrary to what our low-dimensional intuition might suggest, the union of two truncated cones is far from being convex, even when the radius at the point of intersection of both cones is very close to the maximum radius of both truncated cones. The construction is defined precisely and is shown to be far from convex in Section 3.

The proof of Theorem 1.1 is completed in Section 4.1, where we show that with high probability the line segment joining two points drawn uniformly at random from the union of two truncated cones is contained within the body. We then build on this result in Section 4.2 to show that the convex hull of $m \geq 2$ points drawn uniformly from that body is also contained within the body with high probability.

1.2 Discussion

Testing convexity efficiently

Our results do not rule out the possibility that convexity of high-dimensional sets can be tested with a number of queries to random and membership oracles that is polynomial in n and $1/\epsilon$, but they do show that new algorithmic techniques that go beyond convex hull testing are required if such an efficient convexity tester exists. To determine which additional techniques might be useful in obtaining such an efficient convexity tester (or ruling out their existence), it might be instructive to point out that the body constructed in Section 3 is in fact very easy to distinguish from convex bodies. One way to do this is to notice that the union of truncated cones has poor *expansion*: if we take a random walk from a point within one of the two truncated cones, with high probability it will remain within the same truncated cone. By contrast, a random walk in a convex body quickly converges to a distribution that is close to uniform in the body. Is it possible to efficiently test if an unknown body is expanding or far from it? And is it also possible to efficiently distinguish convex sets from expanding sets that are far from convex? Affirmative answers to both of these questions – for any reasonable formalization of the expansion testing problem – would likely lead to a new efficient tester for convexity; the question of testing expansion of high-dimensional sets also appears to be worth studying for independent interest as well.

Testing convexity over the Gaussian distribution

There is another formalization of the geometric convexity testing problem in which we measure the distance to convexity in terms of of the Gaussian distribution on $\mathbb{R}^n$. Chen, Freilich, Servedio, and Sun [10] studied *sample-based* testers for convexity in this model – testers that have access to the membership oracle but can only observe its responses to points drawn from the Gaussian distribution. They showed that all such sample-based testers for convexity have exponential sample complexity. Could the construction introduced in this paper be extended to show a similar bound for the query complexity of a wider class of testers in the same setting? Such results do not follow immediately from the current work since the argument showing that the union of truncated cones is far from convex does not hold in the Gaussian distribution setting.

Testing convex functions

Another problem that has received a considerable amount of attention in the property testing literature, starting with the work of Parnas, Ron, and Rubinfeld [17], is that of testing the convexity of *functions* [8, 4, 9, 3]. There is a close connection between convexity of sets and convexity of functions. Namely, a function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if and only if its epigraph is a convex set in $\mathbb{R}^{n+1}$. The definitions of distance to convexity, however, make the problems of testing convex functions and testing convex sets quite different in general. Nonetheless, as Berman, Raskhodnikova, and Yaroslavtsev [8] pointed out, the two problems are closely connected when we consider the testing of convex functions under the ℓ_1 norm, and it would be interesting to see if the techniques or results introduced here could yield any progress on the problem of testing convex functions with a polynomial number of queries. (See [8, 22] for more details on this problem.)

2 Preliminaries

We use standard notions and results regarding high-dimensional convex sets. For general introductions to the topic and to algorithmic implications, see [1, 15, 16, 14, 21].

2.1 Convex bodies and slices

The *distance* between two bodies $A, B \in \mathbb{R}^n$ is defined to be

$$\mathrm{dist}(A, B) = \mathrm{Vol}(A \triangle B) = \mathrm{Vol}(A \setminus B) + \mathrm{Vol}(B \setminus A),$$

the measure of the symmetric difference of the two bodies. We will repeatedly use the following simple lower bound on the distance of two bodies.

▶ **Proposition 2.1.** *The distance between two bodies $A, B \subset \mathbb{R}^n$ is bounded below by*

$$\mathrm{dist}(A, B) \geq \max\big\{\mathrm{Vol}(A) - \mathrm{Vol}(B), \mathrm{Vol}(B) - \mathrm{Vol}(A)\big\}.$$

Furthermore, equality holds whenever $A \subseteq B$ or $B \subseteq A$.

Proof. The distance between A and B is bounded below by

$$\mathrm{dist}(A, B) = \mathrm{Vol}(A \setminus B) + \mathrm{Vol}(B \setminus A) \geq \max\{\mathrm{Vol}(A \setminus B), \mathrm{Vol}(B \setminus A)\}.$$

The lower bound then follows from the observation that $\mathrm{Vol}(A \setminus B) = \mathrm{Vol}(A) - \mathrm{Vol}(A \cap B) \geq \mathrm{Vol}(A) - \mathrm{Vol}(B)$ and, similarly, that $\mathrm{Vol}(B \setminus A) \geq \mathrm{Vol}(B) - \mathrm{Vol}(A)$. Finally, when $A \subseteq B$, then $\mathrm{Vol}(A \setminus B) = 0$ and $\mathrm{Vol}(B \setminus A) = \mathrm{Vol}(B) - \mathrm{Vol}(A)$, as $\mathrm{Vol}(A \cap B) = \mathrm{Vol}(A)$ so equality holds. Similarly, equality also holds when $B \subseteq A$. ◀

Much of our analysis in Section 3 is concerned with various slices of a high-dimensional body. To make this notion precise, for each $t \in \mathbb{R}$ we define

$$H_t = \{x \in \mathbb{R}^n : x_1 = t\}$$

to be the hyperplane of all points with first coordinate value t. The corresponding halfspaces are denoted by $H_{\leq t} = \{x \in \mathbb{R}^n : x_1 \leq t\}$ and $H_{\geq t} = \{x \in \mathbb{R}^n : x_1 \geq t\}$. The t-th *slice* of a body $A \subset \mathbb{R}^n$ is

$$A_t = A \cap H_t = \{x \in A : x_1 = t\}.$$

For $t_1 \leq t_2 \in \mathbb{R}$, we also define $A_{[t_1,t_2]} = A \cap H_{\geq t_1} \cap H_{\leq t_2}$ to be the set of points in A with first coordinate between t_1 and t_2.

A fundamental property of the slices of a convex body is that the $(n-1)$-th root of their volumes is a concave function.

▶ **Brunn's Theorem.** *For any convex body $C \subset \mathbb{R}^n$, the function $t \mapsto \mathrm{Vol}_{n-1}(C_t)^{\frac{1}{n-1}}$ is concave on its support.*

2.2 High-dimensional balls and cones

We use $B_n(x, r) = \{y \in \mathbb{R}^n : \|y - x\| \leq r\}$ to denote the ball of radius r around a point $x \in \mathbb{R}^n$. We use $B(r)$ as a shorthand for $B_n(o, r)$, and $B_{n-1}(r)$ for $B_{n-1}(o, r)$, where o is the origin. Similarly, we use $S_n(x, r) = \{y \in \mathbb{R}^n : \|y - x\| = r\}$ to denote the sphere of radius r around a point $x \in \mathbb{R}^n$. We will use the following standard approximation on the volume of the ball.

▶ **Proposition 2.2.** *The volume of ball $B(r) \subset \mathbb{R}^n$ with radius r is*

$$\mathrm{Vol}(B(r)) = r^n \cdot \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} = r^n \cdot \frac{1}{\sqrt{\pi n}}\left(\frac{2\pi e}{n}\right)^{\frac{n}{2}} \cdot (1 + O(n^{-1})),$$

where Γ is Euler's gamma function.

We also use the following standard concentration inequality for high-dimensional balls. (See, e.g., [1].)

▶ **Proposition 2.3.** *Let $x \in \mathbb{R}^n$ be drawn uniformly at random from $B(r)$. Then*

$$\Pr\left[|x_1| \geq \tfrac{r}{100}\right] \leq 2^{-\Omega(n)}.$$

▶ **Definition 2.4.** *Let $H \subset \mathbb{R}^n$ be a hyperplane, $S \subset H$ be an $(n-1)$-dimensional convex body, and $x \in \mathbb{R}^n \setminus H$ be a point. The* cone *with x as vertex and S as base is the convex hull of x with the body S;*

$$\mathrm{cone}(x, S) = \mathrm{conv}(x \cup S).$$

We use the following result on the volume of cones.

▶ **Proposition 2.5.** *Let $H \subset \mathbb{R}^n$ be a hyperplane, $S \subset H$ be an $(n-1)$-dimensional convex body, and $x \in \mathbb{R}^n \setminus H$ be a point at a distance $h = \min_{y \in H} \|x - y\|$ from the hyperplane H. Then the volume of the cone is*

$$\mathrm{Vol}(\mathrm{cone}(x, S)) = \frac{h}{n}\mathrm{Vol}_{n-1}(S).$$

▶ **Definition 2.6.** *A* truncated cone *is the convex hull of two balls $B_{n-1}((t_1, 0 \ldots, 0), r_1) \subset H_{t_1}$ and $B_{n-1}((t_2, 0 \ldots, 0), r_2) \subset H_{t_2}$, $\mathrm{conv}(B_{n-1}((t_1, 0 \ldots, 0), r_1) \cup B_{n-1}((t_2, 0 \ldots, 0), r_2))$.*

3 Union of truncated cones

Theorems 1.1 and 1.2 are both established by analyzing a construction obtained by taking the union of two truncated cones. We describe this construction in Section 3.1. The main technical component of the proofs is in Section 3.2, where we show that the union of truncated cones is far from convex.

3.1 Description of the union of truncated cones D

The body $D \subset \mathbb{R}^n$ that will show the non-robustness of the line and convex hull definition is defined as follows. First, let $d > 0$ be some distance parameter. This distance parameter does not affect the results in the following sections; the reader may fix $d = 1$ for simplicity.

Let B_L, B_R, and B_O be three $(n-1)$-dimensional balls in the hyperplanes H_{-d}, H_d, and H_0, respectively. We define the balls B_L and B_R to have radius 1.1 each and be centered on the points $(-d, 0, 0, \ldots, 0)$ and $(d, 0, 0, \ldots, 0)$, respectively, while B_O has radius 1 and is centered at the origin. Define the body D to be the union of the truncated cones $\mathrm{conv}(B_L, B_O)$ and $\mathrm{conv}(B_O, B_R)$,

$$D = \mathrm{conv}(B_L, B_O) \cup \mathrm{conv}(B_O, B_R).$$

The definition of the body D is illustrated in Figure 1.

One of the basic properties of the body D that we will use in later sections is that a point drawn uniformly at random from D will have a large value in its first coordinate with high probability.

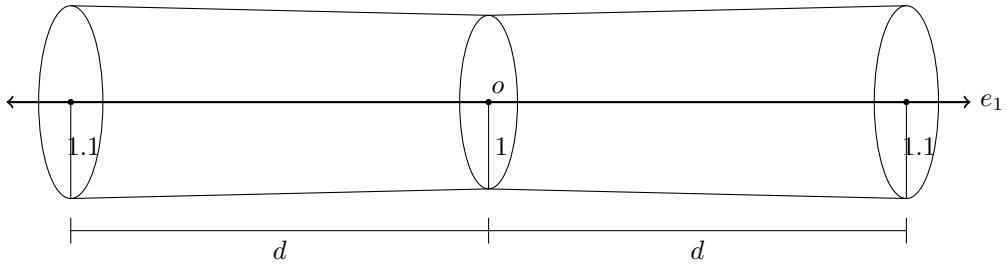


Figure 1 The body D obtained by taking the union of two truncated cones.

▶ **Proposition 3.1.** *Let $x = (x_1, \ldots, x_n)$ be drawn uniformly at random from D. Then*

$$\Pr\left[|x_1| \leq \frac{d}{2}\right] = 2^{-\Omega(n)}.$$

Proof. Using the formula in Proposition 2.5 for the volume of a cone, the volume of the body D is bounded below by

$$\begin{aligned}\mathrm{Vol}(D) = 2\,\mathrm{Vol}(D_{[0,d]}) &= 2\Big(\frac{11d}{n}\,(1.1)^{n-1}\,\mathrm{Vol}_{n-1}\big(B_{n-1}(1)\big) \\ &\qquad - \frac{10d}{n}\,(1)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1))\Big) \\ &\geq \frac{2d}{n}\,(1.1)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1)).\end{aligned}$$

Similarly, the volume of the body $D_{[-\frac{d}{2},\frac{d}{2}]}$ is bounded above by

$$\begin{aligned}\mathrm{Vol}(D_{[-\frac{d}{2},\frac{d}{2}]}) = 2\,\mathrm{Vol}(D_{[0,\frac{d}{2}]}) &= 2\Big(\frac{10.5d}{n}\,(1.05)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1)) \\ &\qquad - \frac{10d}{n}\,(1)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1))\Big) \\ &\leq \frac{21d}{n}\,(1.05)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1)).\end{aligned}$$

Therefore, the probability that the absolute value of the first coordinate of the point is less than or equal to $\frac{d}{2}$ is bounded by

$$\Pr\left[|x_1| \leq \frac{d}{2}\right] = \frac{\mathrm{Vol}(D_{[-\frac{d}{2},\frac{d}{2}]})}{\mathrm{Vol}(D)} \leq \frac{\frac{21d}{n}\,(1.05)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1))}{\frac{2d}{n}\,(1.1)^{n-1}\,\mathrm{Vol}_{n-1}(B_{n-1}(1))} \leq \frac{1}{2^{\Omega(n)}}.$$ ◀

3.2 D is far from convex

In this section, we prove that the body D is $\frac{1}{8}$-far from convex. We prove this in three steps. First, we show that the closest convex body to D must be symmetric about the axis $e_1 = (1, 0, 0, \ldots, 0)$. Next, we prove that if the closest convex body is symmetric about e_1, then it has to be a truncated cone. Finally, we prove that every truncated cone is $\frac{1}{8}$-far from our body D.

3.2.1 A partial converse to Brunn's Theorem

The first step in the proof – showing that the closest convex body to D must be symmetric – uses Brunn's Theorem as well as the following partial converse result.

▶ **Lemma 3.2.** *Let $K \subset \mathbb{R}^n$ be a body such that for each $t \in \mathbb{R}$ the slice K_t is an $(n-1)$ dimensional ball and the function $t \mapsto \mathrm{Vol}_{n-1}(K_t)^{\frac{1}{n-1}}$ is concave on its support. Then K is convex.*

Proof. Assume for the sake of contradiction that K is a non-convex body that satisfies the conditions of the theorem. Then there exist points $x, y \in K$ and $0 \leq \lambda \leq 1$ such that the point $z = \lambda x + (1-\lambda)y \notin K$.

Let $r : \mathbb{R} \to \mathbb{R}$ be the function defined by setting $r(t)$ to be the radius of the $(n-1)$-dimensional ball with volume $\mathrm{Vol}_{n-1}(K_t)$. By applying the formula for volume of the ball from Proposition 2.2, we have that

$$r(t) = \frac{\Gamma(\frac{n-1}{2}+1)^{\frac{1}{n-1}}}{\sqrt{\pi}} \mathrm{Vol}_{n-1}(K_t)^{\frac{1}{n-1}}.$$

Since the function $t \mapsto \mathrm{Vol}_{n-1}(K_t)^{\frac{1}{n-1}}$ is concave, the function r is concave as well.

The concavity of r and the fact that $x, y \in K$ imply that

$$r(z_1) = r(\lambda x_1 + (1-\lambda)y_1) \geq \lambda r(x_1) + (1-\lambda) r(y_1) \geq \lambda \sqrt{\sum_{2 \leq i \leq n} x_i^2} + (1-\lambda)\sqrt{\sum_{2 \leq i \leq n} y_i^2}.$$

The fact that $z \notin K$ also implies that

$$r(z_1) < \sqrt{\sum_{2 \leq i \leq n} z_i^2} = \sqrt{\sum_{2 \leq i \leq n} (\lambda x_i + (1-\lambda)y_i)^2}.$$

But by the convexity of Euclidean norm and Jensen's inequality, $\sqrt{\sum_{2 \leq i \leq n} (\lambda x_i + (1-\lambda)y_i)^2}$ $\leq \lambda \sqrt{\sum_{2 \leq i \leq n} x_i^2} + (1-\lambda)\sqrt{\sum_{2 \leq i \leq n} y_i^2}$ so the last two inequalities yield the desired contradiction. ◀

3.2.2 Symmetry of the closest convex body

A body $K \subset \mathbb{R}^n$ is *symmetric* about $e_1 = (1, 0, 0, \ldots, 0)$ if for every point $x \in K$, all points $y \in \mathbb{R}^n$ that satisfy $x_1 = y_1$ and $\sum_{i=2}^n x_i^2 = \sum_{i=2}^n y_i^2$ are also in K. In other words, a body K is symmetric about e_1 if it is invariant under rotations about the axis e_1. We use a standard symmetrization argument to prove that the closest convex body to D is symmetric about e_1.

▶ **Lemma 3.3.** *There exists a closest convex body to D that is symmetric about e_1.*

Proof. Fix C to be any convex body which minimizes $\mathrm{dist}(D, C)$. C should be contained between H_{-d} and H_d, otherwise we can truncate C and get a convex body closer to D. Let C^s be the body where for every $t \in \mathbb{R}$, the slice C_t^s of the body is an $(n-1)$-dimensional ball centered at $(t, 0, 0, \ldots, 0)$ and has volume $\mathrm{Vol}_{n-1}(C_t^s)$ equal to the volume of the slice C_t of C. By this construction, C^s is symmetric about e_1. To complete the proof, we need to show that it satisfies $\mathrm{dist}(D, C^s) \leq \mathrm{dist}(D, C)$ and that it is convex.

We first establish the inequality $\mathrm{dist}(D, C^s) \leq \mathrm{dist}(D, C)$. The distance between D and C is

$$\mathrm{dist}(D, C) = \int_{t=-d}^{d} \mathrm{dist}(D_t, C_t)\,\mathrm{d}t = \int_{t=-d}^{d} \mathrm{Vol}_{n-1}(D_t \triangle C_t)\,\mathrm{d}t.$$

By Proposition 2.1, for every $t \in \mathbb{R}$, the volume of the symmetric difference between the slices D_t and C_t is bounded below by

$$\mathrm{Vol}_{n-1}(D_t \triangle C_t) \geq \max\Big\{\mathrm{Vol}_{n-1}(C_t) - \mathrm{Vol}_{n-1}(D_t), \mathrm{Vol}_{n-1}(D_t) - \mathrm{Vol}_{n-1}(C_t)\Big\}.$$

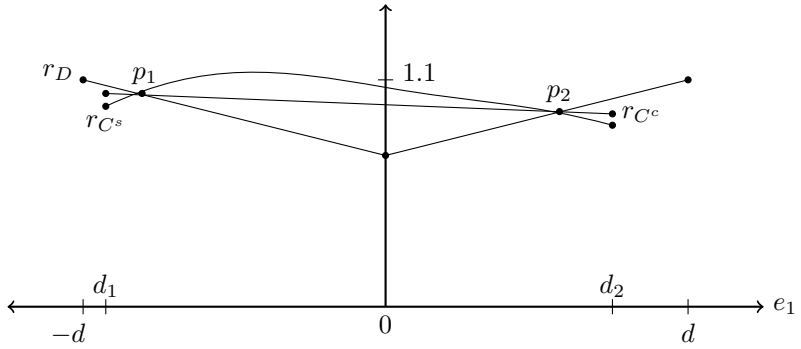


Figure 2 Illustration of the radius functions r_D, r_{C^s}, and r_{C^c} and of the points p_1 and p_2 in the proof of Lemma 3.5.

Since $\mathrm{Vol}_{n-1}(C_t) = \mathrm{Vol}_{n-1}(C_t^s)$, we then obtain

$$\mathrm{dist}(D,C) \geq \int_{t=-d}^{d} \max\left\{\mathrm{Vol}_{n-1}(C_t^s) - \mathrm{Vol}_{n-1}(D_t), \mathrm{Vol}_{n-1}(D_t) - \mathrm{Vol}_{n-1}(C_t^s)\right\} \mathrm{d}t.$$

Both D_t and C_t^s are balls with the same center, so one is a strict subset of the other and so we can apply the equality condition of Proposition 2.1 to obtain

$$\mathrm{dist}(D,C) \geq \int_{t=-d}^{d} \mathrm{Vol}_{n-1}(D_t \triangle C_t^s)\,\mathrm{d}t = \int_{t=-d}^{d} \mathrm{dist}(D_t, C_t^s)\,\mathrm{d}t = \mathrm{dist}(D, C^s),$$

as we wanted to show.

We now complete the proof of the lemma by showing that C^s is convex. From Brunn's Theorem, the function $t \mapsto \mathrm{Vol}_{n-1}(C_t)^{\frac{1}{n-1}}$ is concave on its support. And from the construction we have that $\mathrm{Vol}_{n-1}(C_t^s)^{\frac{1}{n-1}} = \mathrm{Vol}_{n-1}(C_t)^{\frac{1}{n-1}}$. Hence, the function $t \mapsto \mathrm{Vol}_{n-1}(C_t^s)^{\frac{1}{n-1}}$ is also concave on its support and by Lemma 3.2, the body C^s is convex. ◀

3.2.3 The closest convex body is a truncated cone

We now show that the closest symmetric convex body to D is a truncated cone. The proof of this claim uses the following standard result about the separation of convex and concave functions.

▶ **Lemma 3.4.** *Fix any $d_1 \leq d_2 \in \mathbb{R}$. Let $f : [d_1, d_2] \to \mathbb{R}$ be a convex function and $g : [d_1, d_2] \to \mathbb{R}$ be a concave function such that $\forall t \in [d_1, d_2]$, $f(t) \geq g(t)$. Then there exists an affine function $h : [d_1, d_2] \to \mathbb{R}$ such that $g(t) \leq h(t) \leq f(t)$ for all $t \in [d_1, d_2]$.*

Proof. The proof follows from the fact that any two convex sets have a separating hyperplane. Let $S_1 = \{(t,x) : t \in [d_1, d_2], x \geq f(t)\}$ and $S_2 = \{(t,x) : t \in [d_1, d_2], x \leq g(t)\}$. The sets S_1 and S_2 are convex and their separating hyperplane corresponds to the function h. ◀

▶ **Lemma 3.5.** *The closest convex body to D that is symmetric about e_1 is a truncated cone.*

Proof. Let C^s be a convex body that is symmetric about e_1 and minimizes $\mathrm{dist}(D, C^s)$. We will construct a truncated cone C^c that is also symmetric about e_1 and satisfies $\mathrm{dist}(D, C^c) \leq \mathrm{dist}(C^s, D)$.

Define the functions $r_D, r_{C^s} : \mathbb{R} \to \mathbb{R}$ where $r_D(t)$ and $r_{C^s}(t)$ are the radii of the $(n-1)$-dimensional balls D_t and C^s_t, respectively. Let d_1 be the infimum of t for which $r_{C^s}(t) > 0$ and d_2 be the supremum of t for which $r_{C^s}(t) > 0$. Note that $-d \le d_1 < d_2 \le d$ since the body C^s is contained between the hyperplanes H_{-d}, H_d.

We define an affine function $r_{C^c} : [d_1, d_2] \to \mathbb{R}$. We further define C^c to be the body whose slices C^c_t are $(n-1)$-dimensional balls of radius $r_{C^c}(t)$, for $t \in [d_1, d_2]$. Clearly C^c is a truncated cone. We define r_{C^c} differently for different cases mentioned below. In Case 1 we define it directly and in Cases 2, 3 we define two values a_1, a_2. The affine function $r_{C^c} : [d_1, d_2] \to \mathbb{R}$ corresponding to the values a_1, a_2 is defined by the line joining the points $p_1 = (a_1, r_D(a_1)), p_2 = (a_2, r_D(a_2))$. See Figure 2 for an illustration of this construction.

- **Case 1: $\forall t \in (d_1, d_2)$, $r_{C^s}(t) \le r_D(t)$**
  Since r_D is convex and r_{C^s} is concave, from Lemma 3.4, there exists an affine function $r_{C^c} : [d_1, d_2] \to \mathbb{R}$ such that $r_D(t) \ge r_{C^c}(t) \ge r_{C^s}(t)$ for all $t \in [d_1, d_2]$.
- **Case 2: $\forall t \in (d_1, d_2)$, $r_{C^s}(t) > r_D(t)$**
  Let $a_1 = d_1$, $a_2 = d_2$.
- **Case 3:** $\exists t_1, t_2 \in [d_1, d_2]$ such that $r_{C^s}(t_1) \le r_D(t_1)$ and $r_{C^s}(t_2) > r_D(t_2)$
  This case be further divided into three sub-cases.
  - **Case 3a:** $r_{C^s}(d_1) \le r_D(d_1)$ and $r_{C^s}(d_2) \le r_D(d_2)$
    In this case since r_{C^s} is concave and r_D is convex the curves have exactly two points of intersection. Let a_1, a_2 be the values of t where the curves intersect.
  - **Case 3b:** $r_{C^s}(d_1) \le r_D(d_1)$ and $r_{C^s}(d_2) > r_D(d_2)$
    In this case since r_{C^s} is concave and r_D is convex the curves have exactly one point of intersection. Let a_1 be the value of t where the curves intersect and let $a_2 = d_2$.
  - **Case 3c:** $r_{C^s}(d_1) > r_D(d_1)$ and $r_{C^s}(d_2) \le r_D(d_2)$
    In this case since r_{C^s} is concave and r_D is convex the curves have exactly one point of intersection. Let a_2 be the value of t where the curves intersect and let $a_1 = d_1$.

Since the function r_{C^c} is affine, it is also concave and so by Lemma 3.2 the body C^c is convex. To complete the proof, we need to show that $\mathrm{dist}(D, C^c) \le \mathrm{dist}(D, C^s)$ in all three cases.

By definition, the distance between D and C^s is

$$\mathrm{dist}(D, C^s) = \mathrm{Vol}(D \setminus C^s) + \mathrm{Vol}(C^s \setminus D) = \int_{-d}^{d} \mathrm{Vol}_{n-1}(D_t \setminus C^s_t) + \mathrm{Vol}_{n-1}(C^s_t \setminus D_t)\,\mathrm{d}t.$$

For Case 1, since D, C^s, C^c are symmetric about e_1 and $r_D(t) \ge r_{C^c}(t) \ge r_{C^s}(t)$ for every $t \in [d_1, d_2]$,

$$\begin{aligned}
\mathrm{dist}(D, C^s) &= \int_{-d}^{d_1} \mathrm{Vol}_{n-1}(D_t)\,\mathrm{d}t + \int_{d_1}^{d_2} \mathrm{Vol}_{n-1}(D_t) \\
&\qquad\qquad - \mathrm{Vol}_{n-1}(C^s_t)\,\mathrm{d}t + \int_{d_2}^{d} \mathrm{Vol}_{n-1}(D_t)\,\mathrm{d}t \\
&\ge \int_{-d}^{d_1} \mathrm{Vol}_{n-1}(D_t)\,\mathrm{d}t + \int_{d_1}^{d_2} \mathrm{Vol}_{n-1}(D_t) \\
&\qquad\qquad - \mathrm{Vol}_{n-1}(C^c_t)\,\mathrm{d}t + \int_{d_2}^{d} \mathrm{Vol}_{n-1}(D_t)\,\mathrm{d}t \\
&= \mathrm{dist}(D, C^c).
\end{aligned}$$

For Cases 2 and 3, for $t \in (d_1, a_1) \cup (a_2, d_2)$, the ball D_t contains the ball C^s_t. Conversely, for every $t \in [a_1, a_2]$, C^s_t contains D_t. And for $t \in (-d, d_1) \cup (d_2, d)$ the ball C^s_t has zero radius Hence, the distance between D and C^s is

$$\begin{aligned}\operatorname{dist}(D, C^s) = &\int_{d_1}^{a_1} \operatorname{Vol}_{n-1}(D_t) - \operatorname{Vol}_{n-1}(C^s_t)\,\mathrm{d}t + \int_{a_1}^{a_2} \operatorname{Vol}_{n-1}(C^s_t) - \operatorname{Vol}_{n-1}(D_t)\,\mathrm{d}t \\ &+ \int_{a_2}^{d_2} \operatorname{Vol}_{n-1}(D_t) - \operatorname{Vol}_{n-1}(C^s_t)\,\mathrm{d}t + \int_{-d}^{d_1} \operatorname{Vol}_{n-1}(D_t)\,\mathrm{d}t \\ &+ \int_{d_2}^{d} \operatorname{Vol}_{n-1}(D_t)\,\mathrm{d}t.\end{aligned}$$

For every $t \in (d_1, a_1) \cup (a_2, d_2)$, we have that $r_D(t) \geq r_{C^c}(t) \geq r_{C^s}(t)$. And for every $t \in (a_1, a_2)$, we have the reverse inequalities $r_D(t) \leq r_{C^c}(t) \leq r_{C^s}(t)$. Therefore,

$$\begin{aligned}\operatorname{dist}(D, C^s) &\geq \int_{d_1}^{a_1} \operatorname{Vol}_{n-1}(D_t) - \operatorname{Vol}_{n-1}(C^c_t)\,\mathrm{d}t + \int_{a_1}^{a_2} \operatorname{Vol}_{n-1}(C^c_t) - \operatorname{Vol}_{n-1}(D_t)\,\mathrm{d}t \\ &\quad + \int_{a_2}^{d_2} \operatorname{Vol}_{n-1}(D_t) - \operatorname{Vol}_{n-1}(C^c_t)\,\mathrm{d}t + \int_{-d}^{d_1} \operatorname{Vol}_{n-1}(D_t)\,\mathrm{d}t \\ &\quad + \int_{d_2}^{d} \operatorname{Vol}_{n-1}(D_t)\,\mathrm{d}t \\ &= \operatorname{Vol}(D \setminus C^c) + \operatorname{Vol}(C^c \setminus D) = \operatorname{dist}(D, C^c).\end{aligned}$$

◀

3.2.4 Every truncated cone is far from D

As the last step in the proof that D is far from convex, we show that it is far from every truncated cone.

▶ **Lemma 3.6.** *Every truncated cone is $\frac{1}{8}$-far from D.*

Proof. Let C^c be a truncated cone. Without loss of generality let the truncated cone have larger radius towards the left side. We consider the two cases where $\operatorname{Vol}(C^c_{[0,d]}) \leq \frac{1}{2}\operatorname{Vol}(D_{[0,d]})$ and where $\operatorname{Vol}(C^c_{[0,d]}) > \frac{1}{2}\operatorname{Vol}(D_{[0,d]})$ separately.

Case 1: $\operatorname{Vol}(C^c_{[0,d]}) \leq \frac{1}{2}\operatorname{Vol}(D_{[0,d]})$.

In this case, Proposition 2.1 and the case condition yield

$$\operatorname{Vol}(D \triangle C^c) \geq \operatorname{Vol}(D_{[0,d]} \triangle C^c_{[0,d]}) \geq \operatorname{Vol}(D_{[0,d]}) - \operatorname{Vol}(C^c_{[0,d]}) \geq \frac{1}{2}\operatorname{Vol}(D_{[0,d]}) = \frac{1}{4}\operatorname{Vol}(D).$$

Case 2: $\operatorname{Vol}(C^c_{[0,d]}) \geq \frac{1}{2}\operatorname{Vol}(D_{[0,d]})$

In this case, if $C^c_{[-d,-\frac{d}{2}]} = \emptyset$, then from Proposition 3.1

$$\operatorname{Vol}(D \triangle C^c) \geq \operatorname{Vol}(D_{[-d,-\frac{d}{2}]} \triangle C^c_{[-d,-\frac{d}{2}]}) = \operatorname{Vol}(D_{[-d,-\frac{d}{2}]}) \geq \frac{1}{4}\operatorname{Vol}(D).$$

If $C^c_{[-d,-\frac{d}{2}]} \neq \emptyset$, then using the fact that C^c is a truncated cone with larger radius on the left side we get

$$\operatorname{Vol}(C^c_{[-\frac{d}{2},\frac{d}{2}]}) \geq \operatorname{Vol}(C^c_{[0,d]}) \geq \frac{1}{2}\operatorname{Vol}(D_{[0,d]}) \geq \frac{1}{4}\operatorname{Vol}(D).$$

Then Proposition 2.1 implies that

$$\begin{aligned}\mathrm{Vol}(D \triangle C^c) &\geq \mathrm{Vol}(D_{[-\frac{d}{2},\frac{d}{2}]} \triangle C^c_{[-\frac{d}{2},\frac{d}{2}]}) \geq \mathrm{Vol}(C^c_{[-\frac{d}{2},\frac{d}{2}]}) - \mathrm{Vol}(D_{[-\frac{d}{2},\frac{d}{2}]}) \\ &\geq \frac{1}{4}\mathrm{Vol}(D) - \mathrm{Vol}(D_{[-\frac{d}{2},\frac{d}{2}]}).\end{aligned}$$

From Proposition 3.1, we also have that $\mathrm{Vol}(D_{[-\frac{d}{2},\frac{d}{2}]})$ is exponentially smaller than $\mathrm{Vol}(D)$. Hence,

$$\mathrm{Vol}(D \triangle C_C) \geq \frac{1}{4}\mathrm{Vol}(D) - o\big(\mathrm{Vol}(D)\big) \geq \frac{1}{8}\mathrm{Vol}(D). \qquad \blacktriangleleft$$

Putting our last three lemmas together completes the proof of the main result from this section.

▶ **Theorem 3.7.** *The body D is $\frac{1}{8}$-far from convex.*

▶ Remark 3.8. In fact, the above argument shows that D is $(\frac{1}{4} - o(1))$-far from convex. With more careful calculations, it is possible to show that D is $(\frac{1}{2} - o(1))$-far from convex. This result is tight, since the body D is $(\frac{1}{2} - o(1))$-close to the convex body obtained by deleting the right half of D and extending the truncated cone in the left half to d.

4 Proofs of Theorems 1.1 and 1.2

We complete the proofs of Theorems 1.1 and 1.2 in this section. The proof of Theorem 1.1 is completed in Section 4.1, where we show that a line segment connecting two points drawn at random from the body is contained within the body with high probability. In Section 4.2, we generalize this result to show that the convex hull of a set of points picked uniformly at random lies inside the body with high probability.

4.1 Non-robustness of the line characterization

We are now ready to prove Theorem 1.1 by showing that when two points x and y are drawn uniformly at random from D, then with high probability the line segment $\overline{xy}$ that connects x to y is completely contained within D.

▶ **Lemma 4.1.** *When $x, y \in D$ are drawn uniformly at random from D, then the line segment $\overline{xy}$ that joins x and y satisfies*

$$\Pr\left[\overline{xy} \not\subseteq D\right] = 2^{-\Omega(n)}.$$

Proof. Let $x = (\alpha, x_2, \ldots, x_n)$ and y be drawn independently and uniformly at random from D. By the symmetry of D with respect to reflection on the axis e_1, we can assume without loss of generality that $\alpha \leq 0$. Furthermore, since D is symmetric with respect to rotations around e_1, we can also assume that $x_2 \geq 0$ and rest of the $x_i = 0$. Hence, without loss of generality let $x = (\alpha, x_2, 0, 0, \ldots, 0)$ and let $y = (\beta, y_2, \ldots, y_n)$.

If $\beta \leq 0$, then both x and y lie in the same half of D, and that half is a convex set so the line segment $\overline{xy}$ is contained in D.

Consider the case now where $\beta > 0$. By Proposition 3.1, with probability $1 - 2^{-\Omega(n)}$ we have $\alpha \leq -d/2$ and $\beta \geq d/2$. Furthermore, for any given β since $(y_2, \ldots, y_n)$ is uniformly distributed over an $(n-1)$-dimensional ball of radius at most 1.1, from Proposition 2.3 we have that $\Pr[|y_2| \leq \frac{1}{10}] = 1 - 2^{-\Omega(n)}$. In the rest of the proof, assume that all three

inequalities $\alpha \leq -\frac{d}{2}$, $\beta \geq \frac{d}{2}$, and $|y_2| \leq \frac{1}{10}$ hold. We will show that in this case, the line passes through the center slice D_0 and, therefore, the line segment $\overline{xy}$ is contained in D, thus completing the proof of the theorem.

Consider the point $z = \frac{|\alpha|}{|\alpha|+|\beta|}(\beta, y_2, \ldots, y_n) + \frac{|\beta|}{|\alpha|+|\beta|}(\alpha, x_2, 0, 0, \ldots, 0)$. The point z lies in the hyperplane H_0. We want to show that it is contained in the slice D_0 or, equivalently, that $\|z\|^2 \leq 1$. By definition,

$$\begin{aligned}\|z\|^2 &= \left(\frac{1}{|\alpha|+|\beta|}\right)^2 \left(|\alpha| y_2 + |\beta| x_2\right)^2 + \left(\frac{|\alpha|}{|\alpha|+|\beta|}\right)^2 \sum_{i=3}^{n-1} y_i^2 \\ &= \left(\frac{|\beta| x_2}{|\alpha|+|\beta|}\right)^2 + \left(\frac{1}{|\alpha|+|\beta|}\right)^2 2|\alpha||\beta| x_2 y_2 + \left(\frac{|\alpha|}{|\alpha|+|\beta|}\right)^2 \sum_{i=2}^{n} y_i^2.\end{aligned}$$

Since x and y are in D, then $\sum_{i=2}^{n} y_i^2 \leq (1.1)^2$ and $x_2 \leq 1.1$. And we have that $y_2 \leq 0.1$. Substituting these bounds into the above expression, we obtain

$$\begin{aligned}\|z\|^2 &\leq \left(\frac{1.1|\beta|}{|\alpha|+|\beta|}\right)^2 + \left(\frac{1}{|\alpha|+|\beta|}\right)^2 .22|\alpha||\beta| + \left(\frac{1.1|\alpha|}{|\alpha|+|\beta|}\right)^2 \\ &= (1.1)^2 - 2.2|\alpha||\beta| \left(\frac{1}{|\alpha|+|\beta|}\right)^2.\end{aligned}$$

Defining $\delta = |\alpha|/|\beta|$, the above equation simplifies to $\|z\|^2 \leq 1.21 - 2.2\frac{\delta}{(1+\delta)^2}$. Since $|\alpha|$ and $|\beta|$ are both in the range $[\frac{d}{2}, d]$, then $\delta \in [\frac{1}{2}, 2]$. The minimum value of the function $\frac{\delta}{(1+\delta)^2}$ in the interval $[\frac{1}{2}, 2]$ is $\frac{2}{9}$, so $\|z\|^2 \leq 1.21 - 2.2 \cdot \frac{2}{9} \leq 1$. ◀

Theorem 1.1 follows immediately from Theorem 3.7 and Lemma 4.1.

4.2 Non-robustness of the convex hull characterization

In this section, we complete the proof of Theorem 1.2 by combining Lemma 4.1 with the following structural result about the body D.

▶ **Lemma 4.2.** *For any finite set $X \subset D$, if the line connecting any two points $x, y \in X$ satisfies $\overline{xy} \subseteq D$, then*

$$\mathrm{conv}(X) \subseteq D.$$

Proof. We prove the claim by induction on the number of points in X. The base case where $|X| = 2$ is trivially true. For the base case where $|X| = 3$, let $X = \{x, y, z\}$ be any set that satisfies $\overline{xy}, \overline{xz}, \overline{yz} \subseteq D$. We can assume without loss of generality that $x, y \in D_{\geq 0}$. If $z \in D_{\geq 0}$ as well, then $\mathrm{conv}(X) \subseteq D$ since $D_{\geq 0}$ is a convex set. Let us now consider the case where $z \in D_{<0}$. Note that a line joining two points $a \in D_{\leq 0}, b \in D_{\geq 0}$ is contained in the body if and only if $\overline{ab} \cap D_0 \neq \emptyset$. From this observation, we get that $\overline{xz} \cap D_0 \neq \emptyset$ and $\overline{yz} \cap D_0 \neq \emptyset$. Define $x' = \overline{xz} \cap D_0$ and $y' = \overline{yz} \cap D_0$. Let w be any point on the line $\overline{xy}$ and define $w' = \overline{zw} \cap H_0$. Since $w \in \overline{xy}$, we have that $w' \in \overline{x'y'}$. Since $x', y' \in D_0$ and D_0 is convex, we must also have that $w' \in D_0$, and so $\overline{zw} \subseteq D$. Since every point in the convex hull of X is on the line $\overline{zw}$ for some $w \in \overline{xy}$, this means that $\mathrm{conv}(X) \subseteq D$.

For the induction step, we assume that the claim is true for all sets with at most k points for some fixed $k \geq 2$. Fix any set $X \subseteq D$ with $k+1$ elements such that every line $\overline{xy}$ connecting $x, y \in X$ is contained in D. We want to show that $\mathrm{conv}(X) \subseteq D$.

Let $x \in X$ be an element for which there exists $y \in X$ that satisfies $x_1 y_1 \geq 0$, i.e x, y are in the same half of D. Such an element x is guaranteed to exist since $|X| \geq 3$. Without loss of generality, assume $x \in D_{\leq 0}$. Define $X_k = X \setminus \{x\}$. By the induction hypothesis, we must have that $\text{conv}(X_k) \subseteq D$. Furthermore, if every $x' \in \text{conv}(X_k)$ satisfies $\overline{x'x} \subseteq D$, then $\text{conv}(X) = \text{conv}(x \cup X_k) \subseteq D$. To complete the proof, let us now assume that there exists $x' \in \text{conv}(X_k)$ for which $\overline{x'x} \not\subseteq D$ and show that this leads to a contradiction.

Define $X_1 = \{y : y \in X_k \cap D_{\leq 0}\}$ and $X_2 = X_k \setminus X_1$. By our choice of x, $|X_1| \geq 1$ and so $|X_2| \leq k-1$. And since $x' \in \text{conv}(X_k) = \text{conv}(X_1 \cup X_2)$, there exist two points $x'' \in \text{conv}(X_1)$ and $x''' \in \text{conv}(X_2)$ such that $x' \in \overline{x''x'''}$. We have $\overline{xx''} \subseteq D$ as $x, x'' \in D_{\leq 0}$ and $D_{\leq 0}$ is convex. And $\overline{xx'''} \subseteq D$ because $\text{conv}(\{x\} \cup X_2) \subseteq D$ from the induction hypothesis. Finally, since $x'', x''' \in \text{conv}(X_k)$ we also have that $\overline{x''x'''} \subseteq D$. Hence, the three points x, x'', x''' satisfy $\overline{xx''} \subseteq D$, $\overline{xx'''} \subseteq D$, and $\overline{x''x'''} \subseteq D$. Therefore, from the induction hypothesis on the set $\{x, x'', x'''\}$, $\text{conv}(x, x'', x''') \subseteq D$. This implies $\overline{xx'} \subseteq D$, which is a contradiction. Therefore, $\text{conv}(X) = \text{conv}(\{x\} \cup X_K) \subseteq D$. ◀

There exists a small constant $c > 0$ such that if we pick $m = 2^{cn}$ points, X, uniformly at random then the probability that $\forall x, y \in X$, $\overline{xy} \subset D$ is greater than $1 - \frac{1}{2^{\Omega(n)}}$. We get this by applying a union bound on Lemma 4.1. This combined with Lemma 4.2 completes the proof of Theorem 1.2.

References

1 Keith Ball. An elementary introduction to modern convex geometry. In *Flavors of geometry*, chapter 1, pages 1–58. MSRI Publications, 1997.

2 Imre Bárány and Zoltán Füredi. Computing the volume is difficult. *Discrete & Computational Geometry*, 2(4):319–326, 1987. doi:10.1007/BF02187886.

3 Aleksandrs Belovs, Eric Blais, and Abhinav Bommireddi. Testing convexity of functions over finite domains. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA*, pages 2030–2045. SIAM, 2020. doi:10.1137/1.9781611975994.125.

4 Omri Ben-Eliezer. Testing local properties of arrays. In *10th Innovations in Theoretical Computer Science Conference, ITCS*, pages 11:1–11:20, 2019.

5 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. The power and limitations of uniform samples in testing properties of figures. In *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS*, pages 45:1–45:14, 2016.

6 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. In *32nd International Symposium on Computational Geometry, SoCG*, pages 17:1–17:15, 2016.

7 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Tolerant testers of image properties. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 55 of *LIPIcs*, pages 90:1–90:14, 2016.

8 Piotr Berman, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lp-testing. In *Symposium on Theory of Computing, STOC*, pages 164–173, 2014.

9 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings of the 29th Conference on Computational Complexity (CCC)*, pages 309–320, 2014.

10 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*, pages 37:1–37:20, 2017.

11 Ronen Eldan. A polynomial number of random points does not determine the volume of a convex body. *Discret. Comput. Geom.*, 46(1):29–47, 2011.

12 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998. doi:10.1145/285055.285060.

13 Navin Goyal and Luis Rademacher. Learning convex bodies is hard. In *Proceedings of the 22nd Conference on Learning Theory (COLT)*, 2009.

14 Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer, 1988.

15 Peter M. Gruber and Jörg M. Wills, editors. *Handbook of Convex Geometry*. North-Holland, 1993.

16 Jiří Matoušek. *Lectures on discrete geometry*, volume 108. Springer, 2002.

17 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. On testing convexity and submodularity. *SIAM J. Comput.*, 32(5):1158–1184, 2003.

18 Luis Rademacher and Santosh Vempala. Testing geometric convexity. In *Proceedings of the 24th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 469–480, 2004.

19 Sofya Raskhodnikova. Approximate testing of visual properties. In *7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM*, volume 2764 of *Lecture Notes in Computer Science*, pages 370–381. Springer, 2003.

20 Ronitt Rubinfeld and Madhu Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comput.*, 25(2):252–271, 1996. doi:10.1137/S0097539793255151.

21 Santosh Vempala. Recent progress and open problems in algorithmic convex geometry. In *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 42–64, 2010.

22 Grigory Yaroslavtsev. Open problems in L_p-testing. Problem 70 on `sublinear.info`, 2016. URL: `https://sublinear.info/70`.

Distributed Testing of Graph Isomorphism in the CONGEST Model



Reut Levi
Efi Arazi School of Computer Science, The Interdisciplinary Center, Herzliya, Israel
reut.levi1@idc.ac.il



Moti Medina
School of Electrical & Computer Engineering, Ben-Gurion University of the Negev, Beer Sheva, Israel
medinamo@bgu.ac.il

Abstract

In this paper we study the problem of testing graph isomorphism (GI) in the CONGEST distributed model. In this setting we test whether the distributive network, G_U, is isomorphic to G_K which is given as an input to all the nodes in the network, or alternatively, only to a single node.

We first consider the *decision* variant of the problem in which the algorithm should distinguish the case where G_U and G_K are isomorphic from the case where G_U and G_K are not isomorphic. Specifically, if G_U and G_K are not isomorphic then w.h.p. at least one node should output reject and otherwise all nodes should output accept. We provide a randomized algorithm with $O(n)$ rounds for the setting in which G_K is given only to a single node. We prove that for this setting the number of rounds of any deterministic algorithm is $\tilde{\Omega}(n^2)$ rounds, where n denotes the number of nodes, which implies a separation between the randomized and the deterministic complexities of deciding GI. Our algorithm can be adapted to the semi-streaming model, where a single pass is performed and $\tilde{O}(n)$ bits of space are used.

We then consider the *property testing* variant of the problem, where the algorithm is only required to distinguish the case that G_U and G_K are isomorphic from the case that G_U and G_K are *far* from being isomorphic (according to some predetermined distance measure). We show that every (possibly randomized) algorithm, requires $\Omega(D)$ rounds, where D denotes the diameter of the network. This lower bound holds even if all the nodes are given G_K as an input, and even if the message size is unbounded. We provide a randomized algorithm with an almost matching round complexity of $O(D + (\epsilon^{-1} \log n)^2)$ rounds that is suitable for dense graphs (namely, graphs with $\Omega(n^2)$ edges).

We also show that with the same number of rounds it is possible that each node outputs its mapping according to a bijection which is an *approximate* isomorphism.

We conclude with simple simulation arguments that allow us to adapt centralized property testing algorithms and obtain essentially tight algorithms with round complexity $\tilde{O}(D)$ for special families of sparse graphs.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases the CONGEST model, graph isomorphism, distributed property testing, distributed decision, graph algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.19

Category RANDOM

Related Version A full version of the paper is available at `https://arxiv.org/abs/2003.00468`.

Funding *Reut Levi*: This research was supported by the ERC advanced grant 834735.
Moti Medina: This research was supported by the Israel Science Foundation under Grant 867/19.



© Reut Levi and Moti Medina;
licensed under Creative Commons License CC-BY



1 Introduction

Testing graph isomorphism is one of the most fundamental computational problems in graph theory. A pair of graphs G and H are isomorphic if there is a bijection that maps the nodes of G to the nodes of H such that every edge of G is mapped to an edge of H and likewise for non-edges. Currently, it is not known whether there exists an efficient algorithm for this problem and in fact it is one of the few natural problems which is a candidate for being in NP-intermediate, that is, neither in P nor NP-complete. In order to obtain efficient algorithms for this problem, various restrictions and relaxations were considered (e.g. [34, 30]). This problem has been extensively studied also in other computational models such as parallel computation models [29, 41, 10, 32, 46, 35, 24, 9] and in the realm of property testing in which the main complexity measure is the query complexity [17, 43, 25, 42, 38, 3].

In the context of distributive models such as the CONGEST [44] and the LOCAL [40] models, the main complexity measure is the round complexity and the computational power is usually considered to be unbounded. Therefore in these models the complexity of the problem may change dramatically. While there seem to be many sensible settings, one of the simplest settings of the problem for distributive models is to test for isomorphism between the distributed network, G_U, and a known graph, G_K, which is given as an input to all the nodes in the network, or alternatively, only to a subset of the nodes [1]. The requirement from the algorithm is that if G_K and G_U are isomorphic, then with high probability [2] all nodes should output accept and that at least one node should output reject otherwise.

Since the property of being isomorphic to a specific graph is inherently global, intuitively we expect the round complexity to be $\Omega(D)$ where D denotes the diameter of the network (even for the case in which G_K is given as an input to all the nodes in the network). As we show, this intuition is correct even for the LOCAL model, in which there is no bound on the message size. Therefore, in the LOCAL model, it is not possible to improve over the trivial algorithm that collects the entire information on the network at a single node in $O(D)$ rounds and tests for graph isomorphism in a centralized manner. In the CONGEST model, in which the message-size is bounded by $O(\log n)$, where n denotes the number of nodes in the network, implementing this trivial solution may require $O(n^2)$ rounds. This leads to the obvious question whether is it possible to obtain round complexity which is better than $O(n^2)$ in the CONGEST model.

Another interesting question is whether we can obtain better bounds if we relax the decision problem (as considered in the realm of *property testing*) such that the algorithm is only required to distinguish between pairs of graphs which are isomorphic and pairs of graphs which are *far* from being isomorphic (according to some predetermined distance measure).

In this setting we define the problem as follows. Let G_U be the distributed network and let m denote the number of edges in the network or an upper bound on this number. We say that a pair of graphs are ϵ-far from being isomorphic if ϵm edges need to be deleted/inserted in order to make the graphs isomorphic, where m denotes the number of edges in the network. An adversarially chosen node, r, receives as an input the graph G_K and a proximity parameter $\epsilon \in (0, 1)$. The requirement from the algorithm is as follows. If G_K and G_U are isomorphic, then w.h.p. all nodes should output accept. If G_K and G_U are ϵ-far from being isomorphic, then w.h.p. at least one node should output no.

[1] This formulation in which G_K is a parameter falls into the category of massively parameterized problems and is also considered in the setting of property testing [17, 25].

[2] We say that an algorithm succeed with high probability, if it succeeds with probability at least $1 - 1/n^c$ for any constant c (without changing the round complexity asymptotically).

1.1 Our Results

In this section we outline our results. We further elaborate on our results in the next sections. In all that follows, unless explicitly stated otherwise, when we refer to distributed algorithms, we mean in the CONGEST model.

A Decision Algorithm and a Lower Bound for the Decision Problem. For the (exact) decision problem we provide a randomized one-sided error algorithm that runs in $O(n)$ rounds and succeeds with high probability (see Theorem 15). The algorithm works even for the setting in which G_K is given only to a single node (that may be chosen adversely). For this setting we prove that any deterministic algorithm requires $\tilde{\Omega}(n^2)$ rounds, which implies a separation between the randomized and the deterministic complexity of the decision problem (see Theorem 17 in Appendix A). We note that our algorithm can be adapted to the semi-streaming model [15] in which it uses $\tilde{O}(n)$ bits of space and performs only one pass (see Theorem 16).

A Lower Bound for the Property Testing Variant. For property testing algorithms we show that even under this relaxation $\Omega(D)$ rounds are necessary even for constant ϵ and constant error probability. This lower bound holds even in the LOCAL model, for two-sided error algorithms, and even if all the nodes receive G_K as an input (see Theorem 18 in Appendix A). It also holds for dense graphs, namely, when $m = \Theta(n^2)$ and for sparse graphs, that is when $m = \Theta(n)$.

A Property Testing Algorithm and Computation of Approximate Isomorphism. We provide a distributed two-sided error property testing algorithm that runs in $O(D + (\epsilon^{-1} \cdot \log n)^2)$ rounds and succeeds with high probability for the case that $m = \Theta(n^2)$, implying that our result is tight up to an additive term of $(\epsilon^{-1} \cdot \log n)^2$ (see Theorem 1). This algorithm works even in the setting in which G_K is given only to a single node. We note that the graphs that are constructed for the lower bound for the exact variant are dense and have a constant diameter. Therefore for these graphs, the property testing algorithm runs in only $O((\epsilon^{-1} \cdot \log n)^2)$ rounds (while the decision algorithm runs in $O(n)$ rounds).

If G_K is given to all the nodes and the graphs are indeed isomorphic then we show that we can also approximately recover the isomorphism with the same round complexity as of testing. Specifically, each node v outputs $g(v)$ where g is a bijection such that the graph $g(G_U)$, namely the graph in which we re-name the nodes according to g, is ϵ-close to G_K.

Simulation Arguments and their application to special families of sparse graphs. Finally, we show, by simple simulation arguments, that it is possible to obtain essentially tight algorithms with $\tilde{O}(D)$ round complexity for special families of sparse graphs by adapting centralized property testing algorithms. In particular, these algorithms apply for bounded-degree minor-free graphs and general outerplanar graphs (see Appendix B) .

1.2 The Decision Algorithm

As described above, a naive approach for testing isomorphism to G_K is to gather the entire information on the network at a single node and then to test for isomorphism in a centralized manner. By the brute-force approach, we may go over all possible bijections between the nodes of the graphs and test for equality between the corresponding graphs. Our algorithm follows this approach with the difference that it only gathers a compressed version of the

network as in the algorithm of Abboud et al. [2] for the Identical Subgraph Detection problem. The idea of their algorithm is to reduce the problem of testing if two graphs are equal to the problem of testing equality between a pair of binary strings. From the fact that the test for equality has a one-sided error, namely it never rejects identical graphs, it follows that our algorithm never rejects isomorphic graphs. To ensure that our algorithm is sound we amplify the success probability of the equality test by repeating the test $\Theta(n)$ times and, as a result, obtain a total round complexity of $O(n)$.

1.3 A Lower Bound for the Decision Problem

We reduce Set-Equality to the problem of deciding isomorphism in the setting in which only a single node receives G_K as an input (as it is the case for our upper bound). The idea is to construct a graph $G_{x,y}$ over n nodes for every pair of strings $x, y \in \{0,1\}^k$ where $k = \Theta(n^2)$ such that $G_{x,y}$ is isomorphic to $G_{x',y'}$ if and only if $x = x'$ and $y = y'$. Let x and y denote the input of Alice and Bob, respectively. In the reduction, G_K is known to Alice and is taken to be $G_{x,x}$. Alice and Bob simulate the distributed algorithm on the graph $G_{x,y}$, which by construction is isomorphic to $G_{x,x}$ if and only if $x = y$, as desired. This reduction yields a lower bound of $\Omega(n^2/\log n)$ rounds for any deterministic algorithm.

1.4 A High-Level Description of the Property Testing Algorithm

Our algorithm closely follows the approach taken by Fischer and Matsliah [17] for testing graph isomorphism in the dense-graph model [26] with two sided-error. However, in order to obtain a round complexity which only depends poly-logarithmically in n (rather than a dependency of $\tilde{O}(\sqrt{n})$ as the query complexity in [17]), we need to diverge from their approach as described next.

1.4.1 The Algorithm of Fischer-Matsliah

The algorithm of Fischer-Matsliah receives as an input a graph G_K (over n vertices), referred to as the known graph, and an oracle access to the adjacency matrix of a graph G_U, referred to as the unknown graph. It also receives the proximity parameter ϵ. The algorithm begins with picking, u.a.r., a sequence of $s = \text{poly}(\epsilon^{-1}, \log(n))$ nodes from the unknown graph. The selection of these nodes induces labels for each node in the graph as follows. The label of each node v is a string of s bits where the i-th bit indicates whether v is a neighbor of the i-th node in the sequence. This labeling scheme guarantees that, with high probability, only "similar" nodes, that is, nodes with similar sets of neighbors, might have identical labels. It is not hard to see that if the graphs are isomorphic, then given that we managed to map the nodes in the sequence according to the isomorphism, both graphs should have the same frequency over labels. More surprisingly, it is shown by Fischer and Matsliah that if the nodes in the sequence are mapped according to the isomorphism then it is possible to extend this mapping on-the-fly and obtain, roughly speaking, an approximate isomorphism. In particular, they showed that as long as each node in the graph is mapped to a node with the same label in the other graph (with respect to the mapped sequence), then the obtained function is close to being an isomorphism. This is due to the fact that nodes which are too "different" are likely to have different labels and the fact that similar nodes are exchangeable. Given a candidate for the approximate isomorphism, the problem is then reduced to testing closeness of graphs. Therefore, if the graphs are isomorphic then by going over all possible mappings of the selected sequence (there are only quasi-polynomially many ways to map

these nodes) and for each such partial mapping obtaining a candidate by extending the partial mapping as described above, one should be able to obtain a candidate that passes the test, namely a function, f, which is close to being an isomorphism. On the other hand, if the graphs are far from being isomorphic then by definition any bijection gives two graphs which are far from each other. Therefore, these two cases can be distinguished by approximating the Hamming distance of the corresponding adjacency-matrices. In turn, this can be done by selecting random locations (that is, potential edges) and checking the values of both matrices in these locations. Since constructing f entirely would be too costly, one needs to be able to generate f on-the-fly. A crucial point is that its generation can not depend on the selection of the random locations. In other words, its generation should be query-order-oblivious. To this end, in the algorithm of Fischer-Matsliah they first test if the distributions over the labels are close. If so, they can safely generate f on-the-fly while ensuring that there is only little dependency between f and the queries the algorithm makes to f. This is done by simply mapping a node v to a random node in G_K that have the same label as v. [3] The query complexity of testing identity of distributions, which is $\Theta(\sqrt{n})$ [4, 45, 28, 5], dominates the query complexity of the algorithm [4]. As shown in [17], in the centralized setting this algorithm is essentially tight.

1.4.2 Our Algorithm

In the CONGEST model, by straight-forward simulation arguments it follows that one can simulate the algorithm of Fischer-Matsliah in $\tilde{O}(D + \sqrt{n})$ rounds by collecting the answers to the queries of the algorithm at a single node and simulating the centralized algorithm (see Claim 19). A crucial observation for improving this bound is that nodes that have the same label also have at least one neighbor in common (with the only exception of the all-zero label), therefore they can be coordinated by one of their common neighbors. Moreover, it is possible to obtain both samples and access to the frequencies of the labels via these coordinators. Our algorithm proceeds as follows. As in [17] a sequence C of s random nodes is selected and is sent to the entire network (in $O(D)$ rounds). Each node figures out its label and broadcasts this label to its neighbors. The node r, that received G_K as an input, selects a random set of potential edges $(i_1, j_1), \ldots, (i_k, j_k)$, where $k = \text{poly}(\log n, \epsilon^{-1})$. It then broadcasts this set to the entire network. For each potential edge, the information whether it is an actual edge in the network is sent to r. For each label of a node in $I = \{i_1, j_1, \ldots, i_k, j_k\}$, the corresponding coordinator sends to r the frequency of this label. From this point the rest of the computation is done centrally at r. We say that a sequence, P, of nodes in G_K is *good* with respect to C and I if it induces the same frequency of labels as C when restricted to labels of nodes in I. The node r goes over all possible mappings of C to the known graph and looks for good sequences (with respect to C and I). For every good sequence, P, r generates a function f on-the-fly: on query v, f maps v to a random node in V_K which is still unmatched and has the same label as v (w.r.t. P). As we show, from the fact that the sequence is good it follows that f is query-order-oblivious. Let $f(G_U)$ denote the graph obtained from G_U after applying f on V_K. As in the algorithm of Fischer-Matsliah, if the graphs are isomorphic and the sequence P is the mapping of C according to the isomorphism, then $f(G_U)$ is guaranteed

[3] The little dependency between f and the queries that the algorithm makes to f comes from the fact that the frequencies of labels of the two graphs are not necessarily identical as they are only guaranteed to be close (w.h.p.).

[4] In fact, in [17] the failure probability of the test for testing identity of distributions needs to be negligible and consequently the query complexity of this test is $\tilde{\Theta}(\sqrt{n})$.

to be (w.h.p.) ϵ-close to G_K. The set of potential edges is then used to approximate the distance between G_K and $f(G_U)$. This allows us to obtain a significant improvement in the round complexity (over the straight-forward simulation), in terms of n, from $\tilde{O}(\sqrt{n})$ to $O(\log^2 n)$.

1.5 High-level Approach for Computing an Approximate Isomorphism

As described in the previous section, if the graphs are isomorphic then w.h.p. the algorithm finds a sequence P that corresponds to a bijection f such that $f(G_U)$ is guaranteed to be (w.h.p.) ϵ-close to G_K. The algorithm accesses f only on a small set of random locations. It is tempting to try to output $f(v)$ for every v in the network. Assume now that every node in the network knows G_K. If the sequence P is indeed the mapping of C according to an isomorphism then the following naive approach should work. Each coordinator can independently map to V_K the nodes that are assigned to it according to their labels. However, it might be the case that P is not the mapping of C according to any isomorphism (although it passed the test). In particular it might be that it is not good with respect to C and V_U (recall that P is good w.r.t. C and I). In this case we may want the coordinators of the nodes to be coordinated such that they exchange the mapping of nodes with "overflow" and "underflow" labels (that is, labels that are more/less frequent in G_U than in G_K, respectively). Since there might be $O(n)$ labels, such coordination might cause too much congestion. To this end we cluster the labels according to their most significant bit and assign a single coordinator to each cluster. Since there are only $\text{poly}(\epsilon^{-1}, \log(n))$ many clusters, these coordinators can coordinate without causing too much congestion as long as they do not need to communicate too much information. The naive approach of sending the IDs of the nodes that are still un-matched is too costly as there might be a small but a constant fraction of such nodes. Alternatively, we show how the matching of these nodes can be coordinated such that each coordinator sends only $O(\log n)$ bits. In particular, each coordinator sends a number that indicates by how much it is "under-flowing" or "overflowing". Each coordinator with an under-flow reserve a set of nodes in V_K that are reserved to be the image of nodes that are assigned to other coordinators. The set of reserved nodes can be determined by the above-mentioned numbers (i.e., the numbers which indicate the under-flows and over-flows of the coordinators). Moreover, we show that each coordinator with an over-flow can determine which nodes are reserved to be the image of its over-flowing nodes. We prove that the resulting mapping, g, is close enough to f in the sense that the distance between $g(G_U)$ and $f(G_U)$ is small. We prove the latter by coupling g and f and showing that they agree on the mapping of most nodes as long as these nodes have enough neighbors (and if they do not, then their contribution to the distance between $g(G_U)$ and $f(G_U)$ has to be small).

1.6 A Lower Bound for the Property Testing Variant

We prove that for any D there exists a family of graphs with diameter $\Theta(D)$ such that any distributed two-sided error property testing algorithm for testing isomorphism on this family of graphs requires $\Omega(D)$ rounds. In the construction we start with a pair of graphs G_1 and G_2 that have diameter $O(D)$ which are far from being isomorphic. The graph G_U is then defined to be composed of G_1 and G_2 and a path of length $\Theta(D)$ that connects the two graphs. Roughly speaking, the idea is to argue that for round complexity which is at most D/c, where c is some absolute constant, the nodes in G_U which belong to the side of G_1 cannot distinguish the case in which the network is composed of two graphs which are isomorphic to G_1 (connected by a path). Likewise for the nodes that belong to the side of

G_2 (that cannot distinguish the case in which the network is composed of two graphs which are isomorphic to G_2). It then follows that the algorithm must err. In the detailed proof, which appears in the appendix, there are some technicalities that need to be addressed in order to prove that the above argument still holds when the nodes may use randomness, port numbers and IDs.

1.7 Related work

In this section we overview results in distributed decision and property testing in the CONGEST model. We also overview related results in centralized property testing.

Distributed Decision. There is a large body of algorithms and lower bounds for the *subgraph detection problem*: given a fixed graph H, and an input graph G, the task is to decide whether G contains a subgraph which is isomorphic to H. The subgraphs considered include: paths [36], cycles [36, 19, 12], triangles [31, 1, 8], cliques [11, 12, 6]. Abboud et al. [2, Sec. 6.2] considered the *identical subgraph detection problem*. In this problem the graph's nodes are partitioned into two equal sets. The task is to decide whether the induced graphs on these two sets are identical w.r.t. to a fixed mapping between the nodes of these two sets. They showed an $\Omega(n^2)$ lower bound on the number of rounds of any deterministic algorithm and a randomized algorithm that performs $O(D)$ rounds which succeeds w.h.p.

Distributed Property Testing for Graph Problems. Distributed property testing was initiated by Censor-Hillel et al. [7]. In particular, they designed and analyzed distributed property testing algorithms for: triangle-freeness, cycle-freeness, and bipartiteness. They also proved a logarithmic lower bound for the latter two properties. While they mainly focus on the bounded degree model and the general model they also studied the dense model[5]. In this model they showed that for a certain class of problems, any centralized property testing algorithm can be emulated in the distributed model such that number of rounds is q^2 where q denotes the number of queries made by the centralized tester. Fraigniaud et al. [23] studied distributed property testing of excluded subgraphs of size 4 and 5. Since the appearance of the above papers, there was a fruitful line of research in distributed property testing for various properties, mainly focusing on properties of whether a graph excludes a fixed sub-graph [22, 20, 21, 14, 13, 18]. Other problems on graphs such as testing planarity, and testing the conductance were studied in [39, 16], respectively.

Centralized Property Testing. Fischer and Matsliah [17] studied the graph isomorphism problem in the dense-graph model [26].[6] They considered four variations of the Graph Isomorphism testing problem: (1) one-sided error, where one of the graphs is known, and there is a query access to the graph which is tested, i.e., the tested graph is "unknown", (2) one-sided error, where there is a query access for both graphs, i.e., both graphs are unknown, (3) two-sided error, where one graph is known, (4) two sided error, where both graphs are unknown. For the first three variants Fischer and Matsliah [17] showed (almost)

[5] In centralized property testing, these models describe the distance measure as well as how the input graph is represented. In the distributive setting the difference between the models is only in the distance measure. The access to the input graph (which is the underlying network) is clearly the same in all models.

[6] In the dense-graph model, a graph G is considered to be ϵ-far from a property Π if the symmetric difference between its edge set to the edge set of any graph in Π is greater than $\epsilon|V(G)|^2$.

matching lower and upper bounds of, respectively: (1) $\tilde{O}(n)$, $\Omega(n)$, (2) $\tilde{O}(n^{3/2})$, $\Omega(n^{3/2})$, and (3) $\tilde{O}(n^{1/2})$, $\Omega(n^{1/2})$, where n is the number of vertices of each input (known or unknown) graph. For the fourth variant they showed an upper-bound of $\tilde{O}(n^{5/4})$ and a lower-bound of $\Omega(n)$. Onak and Sun [43] improved the upper bound of the fourth case to $O(n) \cdot 2^{\tilde{O}\left(\sqrt{\log n}\right)}$ by bypassing the distribution testing reduction that was used by [17]. Property testing of graph isomorphism was also considered in the bounded-degree model [27][7]. Goldreich [25] proved that the query complexity of any property testing algorithm is at least $\tilde{\Omega}(n^{1/2})$, for the variant in which one graph is known, and $\tilde{\Omega}(n^{2/3})$ when both graphs are unknown. Newman and Sohler [42] provide an algorithm for minor-free graphs with degree bounded by $d = O(1)$ (this class includes for example bounded degree planar graphs) whose query complexity is independent of the size of the graph. Moreover, they showed that any property is testable in this class of graphs with the same query complexity. Kusumoto and Yoshida [38], and Babu, Khoury, and Newman [3] considered testing of isomorphism between graphs which are forests and outerplanar in the general model [33][8], respectively. They both proved an upper bound of $\operatorname{poly}\log n$ and a lower bound of $\Omega(\sqrt{\log n})$ was shown in [38]. Moreover, they proved that any graph property is testable on these family of graphs with $\operatorname{poly}\log n$ queries.

2 The Algorithm for Testing Isomorphism in Dense Graphs

In this section, we describe and analyze the distributed algorithm for testing graph isomorphism in dense graphs. We begin with several useful definitions and observations, followed by the listing Algorithm 1 and the proof of its correctness (which follows from Lemma 11 and Lemma 12). Finally we discuss in more details how the algorithm is implemented in the CONGEST model.

We establish the following theorem.

▶ **Theorem 1.** *There exists a distributed two-sided error property testing algorithm for testing isomorphism (of dense graphs) that runs in $O(D + (\epsilon^{-1}\log n)^2)$ rounds in the CONGEST model. The algorithm succeeds with high probability.*

2.1 Definitions and Notation

We shall use the following definitions in our algorithm and in its analysis.

Let G be a graph and let $C = (c_1, \ldots, c_s)$ be a sequence of s nodes from $V(G)$.

▶ **Definition 2** ([17])**.** *For every node $v \in V(G)$, the C-label of v in G, denoted by $\ell_C^G(v)$, is a string of s bits defined as follows:*

$$\ell_C^G(v)_i = 1 \Leftrightarrow c_i \in N_G(v)\,,$$

where $N_G(v)$ denotes the neighbors of v in G.

For the graphs we consider, $G = (V, E)$, we assume $V \subseteq [|V|^c]$ for some absolute constant $c \geq 1$. In particular we assume that there is a total order defined on V. We use the $\triangle$-operator to denote both the symmetric difference between two sets and when applied on

[7] In the bounded-graph model, a graph G with maximum degree d, is considered to be ϵ-far from a property Π if the symmetric difference between its edge set to the edge set of any graph in Π is greater than $\epsilon d|V(G)|$

[8] In the general-graph model, a graph G, is considered to be ϵ-far from a property Π if the symmetric difference between its edge set to the edge set of any graph in Π is greater than $\epsilon|E(G)|$.

graphs it denotes the Hamming distance between the corresponding adjacency matrices. We assume that the columns and rows of the adjacency matrices are ordered according to the order on the vertices.

▶ **Definition 3** ([17]). *For $\beta \in (0,1]$, we say that C is β-separating if for every pair of nodes u, v such that $|\triangle(N(u), N(v))| \geq \beta n$ it holds that u and v have different C-labels in G.*

▶ **Definition 4** (inverse of ℓ_C^G). *For a label $x \in \{0,1\}^s$, define $S_C^G(x) \triangleq \{v \in V(G) : \ell_C^G(v) = x\}$. Namely, $S_C^G(x)$ is the set of nodes in G for which the C-label is x.*

Let G and H be a pair of graphs such that $|V(G)| = |V(H)|$. The following definitions are defined with respect to a pair of sequences of $s \geq 1$ nodes from $V(G)$ and $V(H)$, $C_G = (c_1^G, \ldots, c_s^G)$ and $C_H = (c_1^H, \ldots, c_s^H)$, respectively.

We next define what we mean by saying that the mapping of a function $f : V(G) \to V(H)$ is consistent w.r.t. the labels of C_G and C_H.

▶ **Definition 5.** *For $f : V(G) \to V(H)$ which is a bijection, we say that f is* (C_G, C_H)-label-consistent *if the following holds:*

1. *f maps C_G to C_H: $f(c_i^G) = c_i^H$ for every $i \in [s]$.*
2. *The label of a node and its image is the same: $\ell_{C_G}^G(v) = \ell_{C_H}^H(f(v))$ for every $v \in V(G)$.*

For $f : V(G) \to V(H)$ and a sequence $C = (c_1, \ldots, c_s)$, we define $f(C)$ to denote $(f(c_1), \ldots, f(c_s))$ and $f(G)$ to denote the graph whose nodes are $V(H)$ and its edge set is $\{\{f(u), f(v)\} : \{u, v\} \in E(G)\}$.

We next observe that if G and H are isomorphic then for any sequence C and any function f which is an isomorphism between G and H, f is consistent w.r.t. C and $f(C)$.

▶ **Observation 1.** *If G and H are isomorphic and π is an isomorphism from $V(G)$ to $V(H)$ then for every sequence of nodes , C, from $V(G)$, π is $(C, \pi(C))$-label consistent. In particular, $|S_C^G(x)| = |S_{\pi(C)}^H(x)|$ for every $x \in \{0,1\}^s$.*

If f is not an isomorphism then it might be the case that it is not consistent w.r.t. C and $f(C)$. We next define a weaker notion of consistency which is being maximally-label-consistent.

▶ **Definition 6.** *We say that a function $f : V(G) \to V(H)$ is* maximally (C_G, C_H)-label-consistent *if the following holds:*

1. *f is a bijection.*
2. *f maps C_G to C_H: $f(c_i^G) = c_i^H$ for every $i \in [s]$.*
3. *For every $x \in \{0,1\}^s$ such that $|S_{C_G}^G(x)| = |S_{C_H}^H(x)|$, f maps the elements of $S_{C_G}^G(x)$ to the elements of $S_{C_H}^H(x)$.*

▶ **Definition 7.** *For a pair of graphs G and H, we say that a function, $f : V(G) \to V(H)$,* ϵ-approximates isomorphism *if f is a bijection and $f(G)$ is ϵ-close to H.*

See Figure 1 for illustration for the definitions in this section.

2.2 Distributed Algorithm Description

The listing of the distributed algorithm appears in Algorithm 1. The detailed description of the distributed implementation of Algorithm 1 appears in Section 2.4.

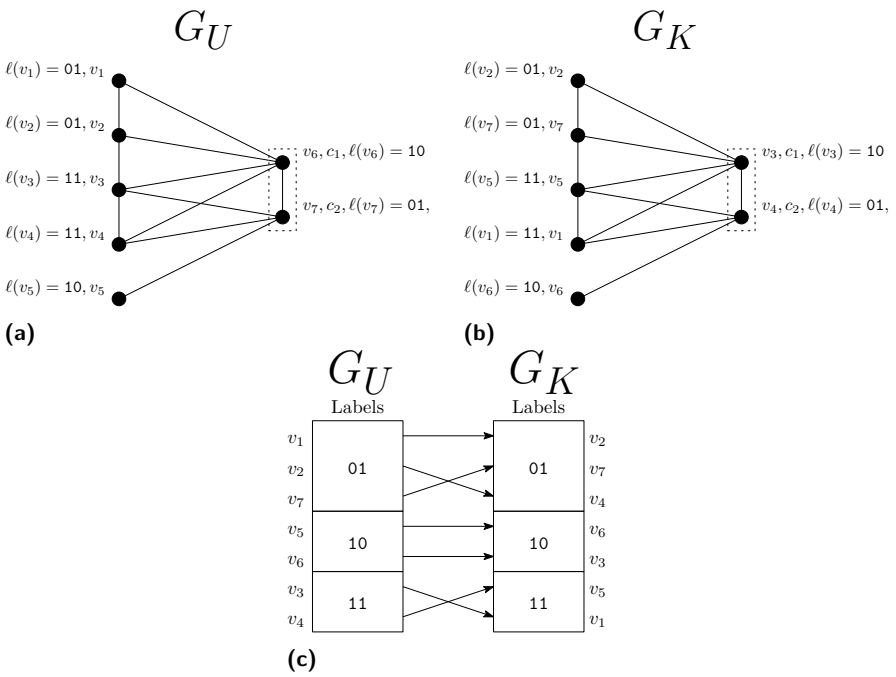


Figure 1 The unknown G_U and known G_K graphs are depicted in Sub-figure 1a, and 1b, respectively. The C sequence consists of the nodes $c_1 = v_6$ and $c_2 = v_7$. Hence, the label of a node v is a binary string $\ell_C^{G_U} = (b_2, b_1) \in \{0,1\}^2$, where $b_i = 1 \Leftrightarrow c_i \in N(v)$, and the same for G_K. In the figures, we removed the super (sub) scripts as they are clear from the context. The induced labels are depicted next to the node number. Note that all the nodes have a neighbor in C, hence the label class 00 is empty. In Sub-figure 1c two tables are depicted: (1) label assignment to the nodes of G_U, and (2) label assignment to the nodes of G_K. The order of which the nodes are presented is according to the (single) isomorphism function between the two graphs. One can observe that the number of nodes per each corresponding label-classes, e.g., a buckets, is the same, as well as buckets which have the same msb bit, e.g., a cluster of labels. A random bijection that preserves the bucket of labels is depicted by the arrows. We show in Thm. 13 that such a bijection yields graphs which are close to being isomorphic.

2.3 Correctness of the Distributed Testing Algorithm

In this subsection we prove the correctness of our algorithm. We begin with a couple of claims and lemmas that we use in our proof. Missing proofs are deferred to Appendix C.

The proof of the following claim appears in [17]. The proof of Lemma 9 can be derived from the proof of Lemma 4.11 in [17] (for the sake of completeness we provide both proofs in the appendix).

▷ Claim 8 ([17]). For $\beta \in (0,1]$ and a sequence, C, of $s = \Theta(\log(n/\delta)/\beta)$ nodes, chosen uniformly at random, C is β-separating with probability at least $1 - \delta$.

▶ **Lemma 9** ([17]). *Let G and H be isomorphic graphs and let π be an isomorphism between them. For any C that is an ϵ-separating sequence of nodes of G and for any f that is $(C, \pi(C))$-label-consistent it holds that $\Delta(f(G), H) \leq \epsilon n^2$.*

The following claim is implied directly from the multiplicative Chernoff's bound.

▷ Claim 10. Let G and H be two graphs such that $V(G) = V(H)$. Then by querying the adjacency-matrices of G and H in $\Theta(\log(1/\delta)/\epsilon)$ random entries it is possible to distinguish between the case that $\Delta(G,H) > \epsilon n^2$ from the case that $\Delta(G,H) \leq \epsilon n^2/2$ with probability at least $1 - \delta$.

Algorithm 1 Testing Isomorphism: The distributed network is $G_U = (V_U, E_U)$.

Input: A "known" graph, $G_K = (V_K, E_K)$ and a proximity parameter ϵ are given as an input to a single node r (may be chosen adversarially).

Output: with high probability, all nodes output yes if G_K is isomorphic to G_U and no otherwise.

1 Compute a BFS tree, T, in G_U rooted at r.

2 Pick, u.a.r., a sequence of $s \triangleq \Theta(\epsilon^{-1} \log |V_U|)$ nodes in G_U. Let $C \triangleq (c_1, \ldots, c_s)$ denote this sequence.

3 Each node $v \in V(G_U)$ computes its label, $\ell_C^{G_U}(v) \in \{0,1\}^s$, according to C and its neighbors in G_U (see Definition 2), and sends this label to its neighbors.

4 The node r picks a sequence of $t \triangleq \Theta(\epsilon^{-1} \log(|V_K|^s))$ pairs of nodes, $A = ((i_1, j_1), \ldots, (i_t, j_t))$, u.a.r. from $V_U \times V_U$. Let $I \triangleq \{i_1, j_1, \ldots, i_t, j_t\}$.

5 For every $e \in A$, r sends e down the BFS tree and learns whether $e \in E_U$ or not. Similarly, for every $v \in I$, it learns $\ell_C^{G_U}(v)$, i.e., the C-label of v in G_U.

6 For each sequence, $P = (p_1, \ldots, p_s)$, of s nodes from V_K, r proceeds as follows:

1. For every $i \in [s]$, verify that $\ell_C^{G_U}(c_i) = \ell_P^{G_K}(p_i)$. If not, then reject P as a candidate and proceed to the next sequence.
2. For every $v \in I$, let ℓ denote $\ell_C^{G_U}(v)$. Check if $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$. If not, then reject P as a candidate and proceed to the next sequence.
3. Pick uniformly at random a function f from the set of all functions that are maximally (C, P)-label-consistent (see Definition 6).
4. Compute the number of edges in A which are non-edges in $f(A)$ and vice-versa. That is, the number of edges $(i_k, j_k) \in A$ such that $(i_k, j_k) \in E_U$ and $(f(i_k), f(j_k)) \notin E_K$, or $(i_k, j_k) \notin E_U$ and $(f(i_k), f(j_k)) \in E_K$). If it is at most $(3\epsilon)|A|/2$ then **return** yes.

If all sequences, P, failed to pass the previous step then **return** no.

▶ **Lemma 11.** *If G_U is isomorphic to G_K then Algorithm 1 accepts with high probability.*

Proof. Assume G_U is isomorphic to G_K and let $\pi : V(G_U) \to V(G_K)$ denote an isomorphism from G_U to G_K. Since the algorithm goes over every sequence P of s nodes from V_K, it also checks $\pi(C)$. By Observation 1, the probability that $\pi(C)$ passes Step 2 is 1. By Claim 8, with high probability, C is $(\epsilon/2)$-separating (see Definition 3). By Lemma 9, if C is $(\epsilon/2)$-separating, then $f(G_U)$ is $(\epsilon/2)$-close to $\pi(G_U) = G_K$. If $f(G_U)$ is $(\epsilon/2)$-close to G_K, then by Claim 10, with high probability $\pi(C)$ passes Step 4. Therefore by the union bound, the algorithm accepts with high probability. ◀

▶ **Lemma 12.** *If G_U is ϵ-far from being isomorphic to G_K then Algorithm 1 rejects with high probability.*

Proof. Assume G_U is ϵ-far from being isomorphic to G_K. We claim that with high probability, any sequence P, fails to pass Step 6. We show this by bounding the probability that a fixed P passes Step 6 and then apply the union bound over all possible sequences. Fix a sequence P and assume that P passes Step 2 (otherwise we are done). Let f be the corresponding f from Step 3 (that is chosen at random). Since G_U is ϵ-far from being isomorphic to G_K, by definition, $\Delta(f(G_U), G_K) \geq \epsilon n^2$. Recall that f is chosen uniformly at random from the set of functions that are maximally (C, P)-label-consistent. By Construction the selection of f is independent of the selection of A. Therefore we can apply Claim 10 on Step 4 as A is a

set of potential edges chosen uniformly at random and, in particular, independently from the random coins in which we use to select f. By Claim 10, P succeeds to pass Step 4 with probability at most $1/|V_K|^{s+c}$ for any absolute constant c. Thus, the lemma follows by a union bound over all possible sequences, as their number is bounded by $|V_K|^s$. ◀

2.4 A Detailed Description of the Distributed Implementation of Algorithm 1

In this section we provide a detailed description of the distributed implementation of our algorithm. We focus on steps for which the implementation is not straightforward and analyze their round complexity. In particular, we focus on Steps 2,6.2,6.3-6.4.

Step 2: Selecting s Nodes u.a.r. In Appendix C.4 we describe a simple procedure to select s nodes uniformly at random (which is a kind of folklore). The round complexity of this procedure is $D + s$.

Step 6.2: Computing the Size of $S_C^{G_U}(\ell)$. Clearly, r can compute $S_P^{G_K}(\ell)$ for any label ℓ as r knows P and G_K. Therefore, in order to describe the implementation of Step 2 it suffices to explain how r can obtain $|S_C^{G_U}(\ell)|$.

We begin with the special case of $\ell = (0, \ldots, 0)$. The nodes in G_U that have this label are nodes that are not adjacent to any one of the nodes in C. Their number can be computed in $O(D)$ rounds by summing it up the BFS tree as follows. Assume w.l.o.g. that every node knows its layer in the BFS-tree. In the first round, every node that is in the last layer (which is also a leaf) sends 1 up to its parent if its C-label is $(0, \ldots, 0)$. In the next round, all nodes in the next layer sum up the received numbers and add 1 if their C-label is $(0, \ldots, 0)$. They send this number up to their parents and so on until we get to the root.

Consider a label ℓ for which at least one bit is 1. Let $\mathrm{msb}(\ell)$ denote the maximum i such that $\ell_i = 1$. Since the node $c_{\mathrm{msb}(\ell)}$ is connected to all nodes whose C-label is ℓ, it can compute their total number (recall that in Step 3 every node sends its C-label to all its neighbors) and send it to the root. Therefore, by a pipelining argument, the root can obtain $|S_C^{G_U}(\ell)|$ for every ℓ which is a C-label of a node in I in $O(D + |I|) = O(D + (\epsilon^{-1} \log n)^2)$ rounds.

Steps 6.3-6.4: Accessing f. Recall that we require from f to be chosen u.a.r. from the set of all functions that are maximally (C, P)-label-consistent (see Definition 6). Recall that f is only evaluated on nodes in I but at the same time its selection has to be independent of A (and I). In particular, the selection of f should be independent of the queries we make to f. To this end, the root verifies the following:

1. In Sub-step 1 of Step 6 it verifies that for the selected sequences, C and P, corresponding nodes have matching labels. Namely, $\ell_C^{G_U}(c_i) = \ell_P^{G_K}(p_i)$ for every $i \in [s]$.
2. In Sub-step 2 of Step 6 it verifies that $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$ for every ℓ which is a C-label of a vertex in I.

If both conditions hold, then it follows that by mapping the nodes in $S_C^{G_U}(\ell)$ to the nodes in $S_P^{G_K}(\ell)$ u.a.r. and independently from the mapping of all other nodes (except for the mapping of C to P which is already determined) for every ℓ such that $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$, we are in fact accessing f which is drawn according to the desired distribution. Therefore the root r simply maps every $v \in I$ to a uniform node $u \in V_K$ such that: (1) $\ell_C^{G_U}(v) = \ell_P^{G_K}(u)$ (2) u is still unmapped (such node always exists). Since r knows G_K and $\ell_C^{G_U}(v)$ for every $v \in I$, it is able to compute $f(v)$ for every $v \in I$, as desired.

Notice that if we evaluated f in the same manner as described above for nodes, v, whose labels, ℓ, are such that $|S_C^{G_U}(\ell)| \neq |S_P^{G_K}(\ell)|$ then the selection of f would not be independent of I. For instance, if $|S_C^{G_U}(\ell)| > |S_P^{G_K}(\ell)|$ then the first queries of nodes in $S_C^{G_U}(\ell)$ will be mapped to $S_P^{G_K}(\ell)$ but then we will run out of nodes in $S_P^{G_K}(\ell)$, implying that the rest of the nodes in $S_C^{G_U}(\ell)$ will be mapped to nodes with a different label than ℓ. Hence, in this case the selection of f will not be oblivious to the order of queries we make to it (and in particular is not independent of I). In the next section we describe a bijection g which can be evaluated on all the nodes in G_U (with the same round complexity as the testing algorithm) and at the same time is close enough to f.

3 Computing an Approximate Isomorphism

In this section we prove the following theorem.

▶ **Theorem 13.** *Let G_U denote the input graph and let G_K be a graph which is isomorphic to G_U and is given as an input to all nodes in the network. There exists a randomized algorithm such that each node in G_U, v, outputs $g(v)$ where $g : V_U \to V_K$ is a bijection such that $g(G_U)$ is ϵ-close to G_K (i.e. g ϵ-approximates isomorphism). The round complexity of the algorithm is $O(D + (\epsilon^{-1} \log n)^2)$. The algorithm succeeds with high probability.*

Proof. The first step of the algorithm is to run Algorithm 1 with the only difference that in Step 6 the root also verifies that $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$ for $\ell = (0, \ldots, 0)$. Since G_K and G_U are isomorphic, by Lemma 11, w.h.p. the algorithm accepts and hence finds P and the corresponding f that pass Step 6. Recall that since f passes the test in Sub-step 6.4 then w.h.p. $f(G_U)$ is ϵ-close to G_K. If every node v could output $f(v)$ then we were done. However, we can not compute f for every node v because for a constant fraction of the nodes its computation might require global information on G_U. Instead, our goal is to output g which is $O(\epsilon)$-close to f and can be computed for every node without causing too much congestion. We next describe g and its computation.

We begin with some notation. Let $L_i \subseteq \{0,1\}^s$ denote the set of labels ℓ for which $\mathrm{msb}(\ell) = i$. Namely, L_i contains all the labels in $\{0,1\}^s$ that can be generated by the following regular expression $0^{i-1}1(0 \cup 1)^{s-i}$. Let Y denote the set of nodes in V_U whose C-label is $(0, \ldots, 0)$. Similarly, let Y' denote the set of nodes in V_K whose P-label is $(0, \ldots, 0)$. For a graph H and a sequence D let $J_D^H(i) \triangleq \cup_{\ell \in L_i} S_D^H(\ell)$, namely, this is the set of all nodes in H whose D-label belongs to L_i. We may refer to $J_D^H(i)$ as the i-th cluster of the graph H w.r.t. D. For $i \in [s]$, define $j_i \triangleq |J_C^{G_U}(i)| - |J_P^{G_K}(i)|$. Namely, j_i is the difference between the sizes of the i-th clusters in both graphs (w.r.t. C and P, respectively).

We next define the set of *reserved* nodes of V_K, denote by R. For each i such that $j_i < 0$, $|j_i|$ nodes from $J_P^{G_K}(i)$ belong to R. Specifically, these are the nodes whose order [9] is the least from the vertices in $J_P^{G_K}(i)$. We consider the order to be the same order as in V_K only that elements in P have the highest order (this is to ensure that none of the elements in P belong to R).

The function g is selected randomly. We next describe the selection of g the outcome of a global random process. Thereafter we describe how this random process can be implemented distributively.

[9] Recall that we assume that there is a total order on V_K which is known to all the nodes in G_U.

Description of g . We describe the selection of g by describing how the nodes in $J_C^{G_U}(i)$ are mapped to V_K, for every $i \in [s]$. We consider three cases, according to the value of j_i.

The first case is when $j_i = 0$. We have the following sub-cases.

1. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$, g matches u.a.r. the elements in $S_C^{G_U}(\ell')$ to the elements in $S_P^{G_K}(\ell')$.
2. The rest of the elements in $J_C^{G_U}(i)$ are matched u.a.r. to the unmatched elements in $J_P^{G_K}(i)$.

Therefore, in this case the elements in $J_C^{G_U}(i)$ are matched only to the elements in $J_P^{G_K}(i)$ and vice versa.

The second case is when $j_i < 0$. We have the following sub-cases.

1. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$ and $S_P^{G_K}(\ell') \cap R = \emptyset$, g matches the elements in $S_C^{G_U}(\ell')$ u.a.r. to the elements in $S_P^{G_K}(\ell')$.
2. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$ and $S_P^{G_K}(\ell') \cap R \neq \emptyset$, g matches u.a.r. a random set of $|S_P^{G_K}(\ell') \setminus R|$ elements from $S_C^{G_U}(\ell')$ to $S_P^{G_K}(\ell') \setminus R$.
3. The rest of the un-matched elements in $J_C^{G_U}(i)$ are mapped u.a.r. to the un-matched elements in $J_P^{G_K}(i) \setminus R$.

Observe that all the elements in $J_C^{G_U}(i)$ are matched to elements in $J_P^{G_K}(i)$ and that the elements that belong to $J_P^{G_K}(i) \cap R$ are still un-matched.

The third case is when $j_i > 0$. We have the following sub-cases.

1. For every $\ell' \in L_i$ such that $|S_C^{G_U}(\ell')| = |S_P^{G_K}(\ell')|$, g matches the elements in $S_C^{G_U}(\ell')$ u.a.r. to the elements in $S_P^{G_K}(\ell')$.
2. The rest of the elements in $J_C^{G_U}(i)$ are matched u.a.r. to the unmatched elements in $J_P^{G_K}(i)$.
3. The remaining $|j_i|$ elements in $J_C^{G_U}(i)$ are matched to the nodes of order $(\sum_{a<i} j_a) + 1$ to $(\sum_{a<i} j_a) + j_i$ in R.

This concludes the description of g for nodes that do not belong to Y. It remains to describe g for nodes that belong to Y. We defer this description to Appendix C.5 as well as the description on how it is computed distributively.

We next describe how g can be computed distributively for nodes that have at least one neighbor in C (namely, nodes that do not belong to Y). Each node $c_i \in C$ is responsible to compute and to send to each node v whose C-label is in L_i the value $g(v)$ (note that c_i and v are necessarily neighbors). As a preliminary step, each node c_i computes $j_i = |J_C^{G_U}(i)| - |J_P^{G_K}(i)|$ and sends (i, j_i) up the BFS tree. Notice that $\sum_{i \in s} j_i = 0$ as $|S_C^{G_U}(\ell)| = |S_P^{G_K}(\ell)|$ for $\ell = (0, \ldots, 0)$ and $|V_U| = |V_K|$. The root sends the set $\{(i, j_i)\}_{i \in s}$ down the BFS tree. By knowing G_K and the set $\{(i, j_i)\}_{i \in s}$, every node c_i can easily compute R. It is not hard to see that this suffices in order to match the elements in $J_C^{G_U}(i)$ to V_K as described above.

By construction it follows that g is a bijection. The bound on the round complexity follows from the bound on the round complexity of Algorithm 1 and the fact that there are only $s = O(\epsilon^{-1} \log n)$ clusters. It remains to prove the following claim.

▷ Claim 14. With high probability $\Delta(g(G_U), G_K) < \epsilon n^2$.

Proof. We observe that both f and g are functions that are chosen randomly. To prove the claim about g we couple g to f and show that they agree on the mapping of most vertices from the set of vertices that have $\Omega(\epsilon n)$ neighbors. This will imply that $\Delta(f(G_U), g(G_U)) = O(\epsilon n^2)$. By Claim 10, since f passed the test in Step 4, w.h.p. $\Delta(f(G_U), G_K) < \epsilon n^2$. We denote this event by E_1. From the fact that w.h.p. E_1 occurs, with combination with the coupling it will follow that w.h.p. $\Delta(g(G_U), G_K) = O(\epsilon n^2)$. Therefore, by setting the proximity parameter to be $\Theta(\epsilon)$ the claim will follow.

Consider the following description of g in terms of f. Let $B = \{\ell \in \{0,1\}^s : |S_C^{G_U}(\ell)| \neq |S_P^{G_K}(\ell)|\}$. For every $v \in V_U$, $g(v) = f(v)$ unless $v \in Y \cup B$ or, $v \notin Y \cup B$ and $f(v) \in R$. In these cases we match v to a node in V_K as listed in the description of g. Observe that under this formulation the distribution of g remains the same. The only difference is that now, for the sake of the analysis, it is coupled to the distribution of f.

It follows that the number of nodes $v \in V_U$ for which $f(v) \neq g(v)$ is at most $|Y|+|R|+|B|$. By Step 2 of Algorithm 1, with high probability it holds that $\sum_{\ell \in B} |S_C^{G_U}(\ell)| \leq \epsilon n$. On the other hand, w.h.p., the number of neighbors of every node in Y is at most ϵn. This implies that the number of nodes in the neighborhood of Y' is at most $O(\epsilon n^2)$ (since otherwise would reject f w.h.p.). Therefore, w.h.p., the contribution of the nodes in Y and Y' to $\Delta(f(G_U), g(G_U))$ is at most $O(\epsilon n^2)$. Thus, w.h.p. $\Delta(f(G_U), g(G_U)) = O(\epsilon n^2)$, as desired. We denote this event by E_2.

Hence, we obtain by union bound over the probability that either $\overline{E_1}$ or $\overline{E_2}$ occur, that w.h.p. $\Delta(g(G_U), G_K) = O(\epsilon n^2)$. By setting the proximity parameter of the algorithm to be ϵ/c for some absolute constant $c > 1$ (which does not affect the asymptotic complexity of the algorithm), we obtain the desired result. ◁

This concludes the proof of the Theorem. ◀

4 Distributed Algorithm for Deciding Isomorphism

In this section we prove the following theorem.

▶ **Theorem 15.** *There exists a randomized distributed algorithm in the CONGEST model that decides if G_K and G_U are isomorphic with high probability. The round complexity of the algorithm is $O(n)$. The algorithm works even if G_K is given only to a single (arbitrary) node.*

Proof. The idea of the algorithm is to go over all possible one-to-one mappings between the nodes of G_K and the nodes of G_U and to test for equality of the corresponding adjacency matrices. The test for equality is performed with very high confidence level in order to ensure that the total error probability is bounded by a small constant. We note that a similar reduction for testing equality also appears in the algorithm of [2, Sec. 6.2] for the Identical Subgraph Detection problem (ISDP).

The first step of our algorithm is to construct a BFS tree and to assign to each node in the network a unique label in $[n]$ where $n \triangleq |V(G_K)|$. This step requires $\Omega(D)$ rounds where D denotes the diameter of G_U (see details on the implementation of this step in the proof of Theorem 13). Consider the adjacency matrix of G_U, M, in which the rows are sorted according to the labels assigned to the nodes. We consider the natural total order on the following set of pairs of nodes $P = \{(i,j) : i \in [n], j \in [n], i < j\}$ in which the pairs are sorted according to the first element and ties are broken according to the second element. Let $\ell(i,j)$ denote the order of the pair (i,j). Each pair $(i,j) \in P$ corresponds to the potential edge between the pair of nodes with labels i and j, respectively. The matrix M can be represented as an integer $s(M)$ where for each $(i,j) \in P$ the $\ell(i,j)$-th lsb (least significant bit) of the binary representation of $s(M)$ indicates whether (i,j) is an edge in the graph. Observe that we can calculate $s(M)$ in D rounds by starting the calculation at the lowest layer of the BFS tree and summing up the outcomes as we go up the tree, layer by layer. The calculation is performed such that each power of two, 2^ℓ, is added (once) if and only if the corresponding edge is present in the graph (i.e. $2^{\ell(i,j)}$ is added if and only if (i,j) is present in the graph).

Since the representation of $s(M)$ requires $O(n^2)$ bits, for our goal we calculate $s(M)$ mod p instead, where p is a prime number in $[n^2]$ chosen uniformly at random. To this end we proceed in the same manner as mentioned above only that before the nodes send up the

tree the outcome of the intermediate sums, they apply the $\mod p$ operation on the outcome. Let $\mathcal{P} = \{p_1, \ldots, p_k\}$ be a multiset of k prime numbers, where $k = \Theta(n)$, each chosen, by a single node v, independently and uniformly at random from the first n^2 primes. [10] The node v broadcasts $\mathcal{P}$ to all the nodes in network in $O(n)$ rounds. Let M' denote the adjacency matrix of G_K. If $M' \neq M$ then the probability that $s(M) \equiv s(M') \mod p$ for a random prime number in $[n^2]$ is at most $1/\Omega(n)$ [37]. Therefore the probability that $s(M) \equiv s(M') \mod p$ every $p \in \mathcal{P}$ is at most $1/\Omega(n^k) = 1/\Omega(n^n)$. Thus we can test with one-sided error if G_U and G_K are equal. The soundness of the equality test is $1 - 1/\Omega(n^n)$. We can apply the equality test for every mapping π between G_K and G_U and return yes if and only if there exists a mapping for which the test accepts. Namely, we go over all possible permutations over the nodes of G_K and for each permutation, π, we perform the equality test between $s(M)$ and $s(\pi(M'))$ where $\pi(M')$ denotes adjacency matrix of G_K after applying the permutation π on the nodes. By the soundness of the equality test and the union bound, the probability that this test returns no when G_K and G_U are not isomorphic is at least $2/3$ (for an appropriate adjustment of the parameters).

By standard pipelining, it is possible to calculate $s(M) \mod p$ for every $p \in \mathcal{P}$ in $O(D+k)$ rounds, therefore the round complexity of the above test is $O(n)$. To verify this observe that in order to execute the above test the only information we need is of G_K and the result of $s(M) \mod p$ for every $p \in \mathcal{P}$. This concludes the proof. ◀

We observe that the above-mentioned algorithm can be adapted to the semi-streaming in a straight-forward way as follows.

▶ **Theorem 16.** *There exists an algorithm in the semi-streaming model that receives a graph G_K over n nodes as an input, where the space for storing G_K is a read-only memory, and a stream of the edges of another graph G_U (according to any order) and decides, with one-sided error, whether G_K and G_U are isomorphic or not. The algorithm performs one-pass and uses $O(n \log n)$ bits of space.*

Proof. Assume w.l.o.g. that the labels of the nodes in G_U and G_K are taken from $[n]$. Otherwise we can re-name then by using a table of size $O(n \log n)$ bits. Let M denote the adjacency matrix of G_U. We first compute $s(M) \mod p$ for every $p \in \mathcal{P}$ as in the proof of Theorem 15, in one-pass, using $O(n \log n)$ bits of space. We then go over all permutations of the nodes of G_K and perform the same computation for the corresponding adjacency matrix. We accept if and only if there exists a permutation π for which $s(M) \equiv s(M') \mod p$ every $p \in \mathcal{P}$, where M' denotes the adjacency matrix of G_K after we permuted the nodes according to π. Observe that we can go over the permutations one by one according to the lexicographical order by using $O(n \log n)$ bits of space. Therefore, the total space the algorithm uses is $O(n \log n)$, as desired. ◀

References

1 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Christoph Lenzen. Fooling views: A new lower bound technique for distributed computations under congestion. *arXiv preprint*, 2017. arXiv:1711.01623.

2 Amir Abboud, Keren Censor-Hillel, Seri Khoury, and Ami Paz. Smaller cuts, higher lower bounds. *CoRR*, abs/1901.01630, 2019. arXiv:1901.01630.

[10] It is well known that for sufficiency large number $x \in \mathbb{N}$ the number of prime numbers that are at most x is $\Theta(x/\log x)$.

3 Jasine Babu, Areej Khoury, and Ilan Newman. Every property of outerplanar graphs is testable. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, volume 60 of *LIPIcs*, pages 21:1–21:19. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.APPROX-RANDOM.2016.21.

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 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing closeness of discrete distributions. *Journal of the ACM (JACM)*, 60(1):1–25, 2013.

6 Matthias Bonne and Keren Censor-Hillel. Distributed detection of cliques in dynamic networks. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece*, pages 132:1–132:15, 2019. doi:10.4230/LIPIcs.ICALP.2019.132.

7 Keren Censor-Hillel, Eldar Fischer, Gregory Schwartzman, and Yadu Vasudev. Fast distributed algorithms for testing graph properties. *Distributed Computing*, 32(1):41–57, 2019. doi:10.1007/s00446-018-0324-8.

8 Yi-Jun Chang, Seth Pettie, and Hengjie Zhang. Distributed triangle detection via expander decomposition. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 821–840. SIAM, 2019.

9 Lin Chen. Parallel graph isomorphism detection with identification matrices. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN)*, pages 105–112. IEEE, 1994.

10 Lin Chen. Graph isomorphism and identification matrices: Parallel algorithms. *IEEE Transactions on Parallel and Distributed Systems*, 7(3):308–319, 1996.

11 Artur Czumaj and Christian Konrad. Detecting cliques in congest networks. *Distributed Computing*, pages 1–11, 2019.

12 Talya Eden, Nimrod Fiat, Orr Fischer, Fabian Kuhn, and Rotem Oshman. Sublinear-time distributed algorithms for detecting small cliques and even cycles. In *33rd International Symposium on Distributed Computing (DISC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

13 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 *Proceedings of the 41st International Symposium on Distributed Computing (DISC)*, pages 15:1–15:30, 2017. doi:10.4230/LIPIcs.DISC.2017.15.

14 Guy Even, Reut Levi, and Moti Medina. Faster and simpler distributed algorithms for testing and correcting graph properties in the congest-model. *CoRR*, abs/1705.04898, 2017. arXiv:1705.04898.

15 Joan Feigenbauma, Sampath Kannanb, Andrew McGregorb, Siddharth Surib, and Jian Zhanga. On graph problems in a semi-streaming model. *Theoretical Computer Science*, 348:207–216, 2005.

16 Hendrik Fichtenberger and Yadu Vasudev. Distributed testing of conductance. *arXiv preprint*, 2017. arXiv:1705.08174.

17 Eldar Fischer and Arie Matsliah. Testing graph isomorphism. *SIAM Journal on Computing*, 38(1):207–225, 2008.

18 Orr Fischer, Shay Gershtein, and Rotem Oshman. On the multiparty communication complexity of testing triangle-freeness. In *Proceedings of the 2017 ACM Symposium on Principles of Distributed Computing (PODC)*, pages 111–120. ACM, 2017.

19 Orr Fischer, Tzlil Gonen, Fabian Kuhn, and Rotem Oshman. Possibilities and impossibilities for distributed subgraph detection. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures*, pages 153–162, 2018.

20 Orr Fischer, Tzlil Gonen, and Rotem Oshman. Distributed property testing for subgraph-freeness revisited. *CoRR*, abs/1705.04033, 2017. arXiv:1705.04033.

21 Pierre Fraigniaud, Pedro Montealegre, Dennis Olivetti, Ivan Rapaport, and Ioan Todinca. Distributed subgraph detection. *CoRR*, abs/1706.03996, 2017. arXiv:1706.03996.

22 Pierre Fraigniaud and Dennis Olivetti. Distributed detection of cycles. *ACM Transactions on Parallel Computing (TOPC)*, 6(3):1–20, 2019.

23 Pierre Fraigniaud, Ivan Rapaport, Ville Salo, and Ioan Todinca. Distributed testing of excluded subgraphs. In *Proceedings of the 30th International Symposium on Distributed Computing (DISC)*, volume 9888 of *LNCS*, pages 342–356. Springer, 2016.

24 Hillel Gazit and J Reif. A randomized parallel algorithm for planar graph isomorphism. In *Proceedings of the second annual ACM symposium on Parallel algorithms and architectures*, pages 210–219, 1990.

25 Oded Goldreich. Testing isomorphism in the bounded-degree graph model. *Electronic Colloquium on Computational Complexity (ECCC)*, 26:102, 2019. URL: https://eccc.weizmann.ac.il/report/2019/102.

26 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *Journal of the ACM (JACM)*, 45(4):653–750, 1998.

27 Oded Goldreich and Dana Ron. Property testing in bounded degree graphs. *Algorithmica*, 32(2):302–343, 2002. doi:10.1007/s00453-001-0078-7.

28 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. In *Studies in Complexity and Cryptography. Miscellanea on the Interplay between Randomness and Computation*, pages 68–75. Springer, 2011.

29 Martin Grohe and Oleg Verbitsky. Testing graph isomorphism in parallel by playing a game. In *International Colloquium on Automata, Languages, and Programming*, pages 3–14. Springer, 2006.

30 John E Hopcroft and Jin-Kue Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 172–184. ACM, 1974.

31 Taisuke Izumi and François Le Gall. Triangle finding and listing in congest networks. In *Proceedings of the ACM Symposium on Principles of Distributed Computing*, pages 381–389, 2017.

32 Joseph Jaja and S Rao Kosaraju. Parallel algorithms for planar graph isomorphism and related problems. *IEEE Transactions on Circuits and Systems*, 35(3):304–311, 1988.

33 Tali Kaufman, Michael Krivelevich, and Dana Ron. Tight bounds for testing bipartiteness in general graphs. *SIAM Journal on Computing*, 33(6):1441–1483, 2004. doi:10.1137/S0097539703436424.

34 Paul J Kelly et al. A congruence theorem for trees. *Pacific Journal of Mathematics*, 7(1):961–968, 1957.

35 Johannes Köbler. On graph isomorphism for restricted graph classes. In *Conference on Computability in Europe*, pages 241–256. Springer, 2006.

36 Janne H Korhonen and Joel Rybicki. Deterministic subgraph detection in broadcast congest. In *21st International Conference on Principles of Distributed Systems (OPODIS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

37 Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.

38 Mitsuru Kusumoto and Yuichi Yoshida. Testing forest-isomorphism in the adjacency list model. In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copenhagen, Denmark, July 8-11, 2014, Proceedings, Part I*, volume 8572 of *Lecture Notes in Computer Science*, pages 763–774. Springer, 2014. doi:10.1007/978-3-662-43948-7_63.

39 Reut Levi, Moti Medina, and Dana Ron. Property testing of planarity in the CONGEST model. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, Egham, United Kingdom, July 23-27, 2018*, pages 347–356, 2018. Full version is available in http://arxiv.org/abs/1805.10657. URL: https://dl.acm.org/citation.cfm?id=3212748.

40 Nathan Linial. Locality in distributed graph algorithms. *SIAM Journal on Computing*, 21(1):193–201, 1992.

41 Eugene M Luks. Parallel algorithms for permutation groups and graph isomorphism. In *27th Annual Symposium on Foundations of Computer Science (sfcs 1986)*, pages 292–302. IEEE, 1986.

42 Ilan Newman and Christian Sohler. Every property of hyperfinite graphs is testable. *SIAM Journal on Computing*, 42(3):1095–1112, 2013. doi:10.1137/120890946.

43 Krzysztof Onak and Xiaorui Sun. The query complexity of graph isomorphism: bypassing distribution testing lower bounds. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 165–171, 2018. doi:10.1145/3188745.3188952.

44 David Peleg. Distributed computing. *SIAM Monographs on discrete mathematics and applications*, 5, 2000.

45 Gregory Valiant and Paul Valiant. The power of linear estimators. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 403–412. IEEE, 2011.

46 Oleg Verbitsky. Planar graphs: Logical complexity and parallel isomorphism tests. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 682–693. Springer, 2007.

47 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. IEEE, 1977.

A Lower Bounds

In this section we establish two lower bounds. The first is for the decision variant of GI, and the second is for the testing variant.

For the decision variant, we prove a near-quadratic lower bound for any deterministic distributed algorithm in the CONGEST model. The second lower bound states that any Isomorphism testing distributed algorithm requires diameter time. This lower bound holds also for randomized algorithms, and in fact holds in the LOCAL model, even if *all* vertices are given as an input the graph G_K.[11]

A.1 An $\Omega(n^2/\log n)$ Lower Bound for Deciding Isomorphism Deterministically

The decision variant of our Isomorphism testing problem is as follows: if G_K and G_U are isomorphic, then all nodes should output yes, while if the graphs are not isomorphic, at least one node should output no.

▶ **Theorem 17.** *Any distributed deterministic algorithm in the CONGEST model for deciding whether G_U is isomorphic to G_K requires $\Omega(n^2/\log n)$ rounds for the setting in which G_K is given as an input to a single node. Moreover, this bound holds even for graphs with constant diameter.*

Proof. We reduce the problem of Set-Equality to the problem of deciding Isomorphism. The reduction is as follows. Alice and Bob each receives as an input a subset of k^2 elements, $A \subseteq \{(x,y) \mid x \in \{a_1^0,\ldots,a_1^{k-1}\}, y \in \{a_2^0,\ldots,a_2^{k-1}\}\}$ and $B \subseteq \{(x,y) \mid x \in \{b_1^0,\ldots,b_1^{k-1}\}, y \in \{b_2^0,\ldots,b_2^{k-1}\}\}$, respectively. According to their inputs they construct a pair of graphs, G_U and G_K, such that $A = B$ if and only if G_U is isomorphic to G_K, as described momentarily.

[11] In the LOCAL model there is no limitation on message size per round per edge. Obviously, a lower bound in the LOCAL model also applies to the CONGEST model.

Since, Set-Equality has a deterministic communication complexity which is linear in size of the universe [37], which is $\Omega(k^2)$ in this case, it follows that any distributed deterministic algorithm in the CONGEST model for deciding whether G_U is isomorphic to G_K requires $\Omega(n^2/\log n)$ rounds.

We now describe the graph G_U which is constructed by Alice and Bob. The set of nodes in Alice's graph is composed of u, u', u'', the subsets $A_1 = \{a_1^0, \ldots, a_1^{k-1}\}$ and $A_2 = \{a_2^0, \ldots, a_2^{k-1}\}$, and the nodes $t_{A_1}, t_{A_2}, t_u^1, t_u^2$ (see Figure 2). The subgraph induced on the nodes in A_1 is the path $a_1^0, \ldots, a_1^{k-1}$. Similarly, the subgraph induced on the nodes in A_2 is the path $a_2^0, \ldots, a_2^{k-1}$. The nodes u and u'' are adjacent to all nodes in A_1 and A_2, respectively. Both nodes u and u'' are adjacent to the node u'. The node u is adjacent to t_u^1 and t_u^2. The nodes a_1^0 and a_2^0 are adjacent to the nodes t_{A_1} and t_{A_2}, respectively. Additionally, the edge (a_1^x, a_2^y) belongs to Alice's subgraph if and only if the corresponding (x, y) element is in A.

The subgraph of Bob is defined similarly only that the nodes $v, v', v'', t_{B_1}, t_{B_2}$ take the role of $u, u', u'', t_{A_1}, t_{A_2}$, respectively, and the subsets B_1 and B_2 take the role of A_1 and A_2, respectively. The edges between B_1 and B_2 are determined by Bob's input B. The other difference is that v is adjacent to three "tails" t_v^1, t_v^2, t_v^3 (whereas u is adjacent to only two, t_v^1, t_v^2).

The subgraphs of Alice and Bob are connected by a single edge (u, v).

The graph G_K is constructed by Alice exactly as the graph G_U, the only difference is that Alice does not know the subset B, instead of B Alice uses A to determine both the edges between A_1 and A_2 and the edges between B_1 and B_2.

Clearly, if A and B are such that $(a_1^i, a_2^j) \in A$ if and only if $(b_1^i, b_2^j) \in B$, then G_U and G_K are isomorphic. We next prove that if this is not the case then G_U and G_K are not isomorphic.

We assume towards contradiction that G_K and G_U are isomorphic and that there exists $(a_1^i, a_2^j) \in A$ such that $(b_1^i, b_2^j) \notin B$ or vice versa. We first observe that it is possible to identify u and v based on the structure of G_U (i.e. regardless to the labels of the nodes) because these are the only nodes that are neighbors to 2 and 3 nodes with degree exactly 1, respectively. Next, we identify u' and v' - these are the nodes that have degree exactly 2 and are adjacent to u and v, receptively. Once we identify u' and v' we can identify u'' and v'' and in turn identify A_2 and B_2. The nodes in A_1 and B_1 are the neighbors of u and v, excluding u' and v', receptively. Finally, we are able to identify a_1^0, a_2^0, b_1^0 and b_2^0 since these are the only nodes that are adjacent to a node of degree exactly one - $t_{A_1}, t_{A_2}, t_{B_1}$ and t_{B_2}, respectively. Once we identify a_1^0 and A_1 we can identify the path $a_1^0, \ldots, a_1^{k-1}$. Likewise for the paths induced on A_2, B_1 and B_2. This implies that there if there exists $(a_1^i, a_2^j) \in A$ such that $(b_1^i, b_2^j) \notin B$ or vice versa then it has to be that the graph G_U, which is constructed with correspondence to the to the pair (A, B), has to be not isomorphic to the graph G_K, which is constructed with correspondence to the to the pair (A, A). This concludes the proof. ◀

A.2 An $\Omega(D)$ lower bound for the Testing Isomorphism Problem

In this section we establish the following theorem.

▶ **Theorem 18.** *For any D there exists a family of graphs with diameter $\Theta(D)$ such that any distributed two-sided error property testing algorithm for testing isomorphism on this family of graphs requires $\Omega(D)$ rounds. This lower bound applies also in the LOCAL model and even if all nodes receive G_K as an input.*

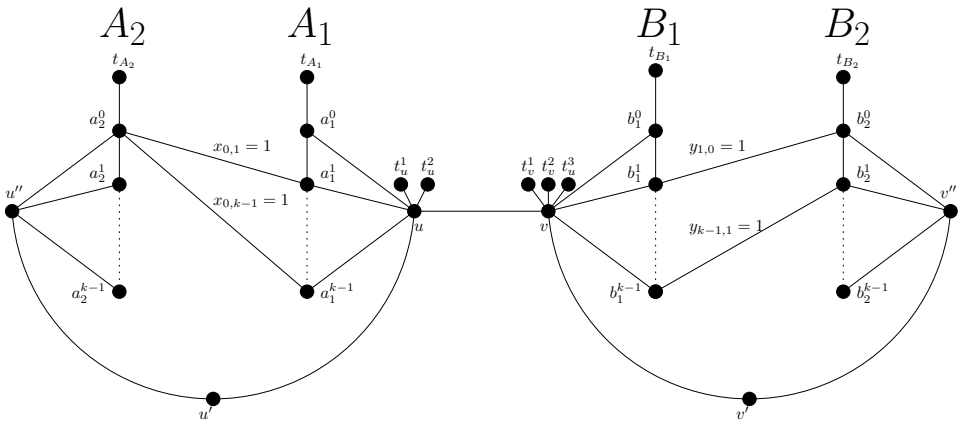


Figure 2 Deciding Isomorphism lower bound construction.

Proof. We define a family of graphs $\mathcal{H}$, where each $H \in \mathcal{H}$ takes the role of the unknown graph G_U. We fix $G_K \in \mathcal{H}$ and require from $\mathcal{H}$ that all graphs in $\mathcal{H} \setminus \{G_K\}$ are ϵ'-far from being isomorphic to G_K. Theorem 18 then follows by applying Yao's principle [47].

Construction of $\mathcal{H}$. We begin our construction with a pair of graphs on n nodes and diameter $\Theta(D)$, G_1 and G_2, such that $|E(G_1)| = (1+\epsilon)|E(G_2)|$. [12] Clearly, G_2 is ϵ-far from being isomorphic to G_1 (since G_1 has $\epsilon|E(G_2)|$ more edges than G_2). Let $\mathcal{H}$ denote a family of graphs where each graph in $\mathcal{H}$, denoted by $G_{i,j}$ for $i,j \in \{1,2\}$, is constructed by using two copies from $\{G_1, G_2\}$. The definition of $G_{i,j}$ is as follows:

1. Let $p \triangleq (p_1, \ldots, p_D)$ denote a path of D nodes.
2. Let v_i and v_j denote arbitrary nodes in G_i and G_j, respectively.
3. Identify, v_i with p_1 and v_j with p_D, to obtain $G_{i,j}$. Observe that $|V(G_{i,j})| = 2n + D - 2 = O(n)$, and that the diameter of $G_{i,j}$ is $\Theta(D)$.

Note that from the reasoning above, for every $G_{i,j}, G_{a,b} \in \mathcal{H}$, where $\{i,j\} \neq \{a,b\}$, $G_{i,j}$ is $\Omega(\epsilon)$-far from being isomorphic to $G_{a,b}$.

We assume towards a contradiction that there is a two-sided tester $\mathcal{A}$ that is correct with probability at least $2/3$ and runs for $D(G_U)/3$ rounds, where $D(G_U)$ denotes the diameter of the distributed network G_U. The lower bound then follows from Yao's principle [47]. We next specify the distribution over the inputs and explicitly apply Yao's principle.

We define a set of inputs to the distributed tester: (1) we fix the known graph G_K to be $G_{1,2}$, (2) the unknown graph G_U is chosen randomly from the distribution $\tilde{\mathcal{H}}$ which is defined as follows: the probability to obtain the graph $G_{1,2}$ is $1/2$ and each of the graphs $G_{1,1}$ and $G_{2,2}$ is obtained with probability $1/4$. Now we consider two cases. The first case is that $\mathcal{A}$ outputs (correctly) yes when G_U is $G_{1,2}$. By the definition of the problem this implies that all nodes output yes. Since the number of rounds of the tester is at most $D/3$, it follows that all nodes in both graphs $G_{1,1}$, and $G_{2,2}$, output yes. To verify this observe that for any node u in either $G_{1,1}$ or $G_{2,2}$ there exists a node in $G_{1,2}$, v, such that the $(D/3)$-hop

[12] Observe that such graphs exist for any D for G_1 and G_2 that are dense. On the other extreme, if we want G_1 and G_2 to be bounded degree trees then it is possible to obtain such graphs for any $D = \Omega(\log n)$.

neighborhoods of v and u are the same (up to labels and port numbers [13]). Thus, in this case, $\mathcal{A}$ errs on $G_{1,1}$ and $G_{2,2}$. In the second case $\mathcal{A}$ outputs no when G_U is $G_{1,2}$. Thus, in both cases $\mathcal{A}$ errs with probability at least $1/2$ when G_U is drawn according to $\tilde{\mathcal{H}}$.

By Yao's principle it is implied that any randomized tester must err with probability at least $1/2$ as well, in contradiction to our assumption that $\mathcal{A}$ is correct with probability of at least $2/3$. ◀

B Simulating Centralized Property Testing Algorithms

In this section we prove the following claim.

▷ Claim 19. Let $\mathcal{A}$ be a centralized property testing algorithm that is allowed to make adjacency queries, incidence queries and degree queries. There exists a distributed algorithm that can simulate $\mathcal{A}$ in $O(D \cdot q)$ rounds if $\mathcal{A}$ is adaptive and in $O(D + q)$ rounds if $\mathcal{A}$ is non-adaptive, where D denotes the diameter of the graph and q denotes the number of queries that $\mathcal{A}$ makes.

Proof. In order to simulate $\mathcal{A}$ on the distributed network we first pick a leader r and construct a BFS tree rooted at r in D rounds. We also assume w.l.o.g. that r knows the size of the network, n. If $\mathcal{A}$ is non-adaptive then r can determine the queries that $\mathcal{A}$ makes by using (only) the knowledge of n and random bits. The root r sends this information to the entire network in $D + q$ rounds and in additional $D + q$ rounds the answers are gathered by pipelining back at r. Once that all the answers are gathered, r can complete the simulation of $\mathcal{A}$. The outcome is then sent to all the nodes in the network. If $\mathcal{A}$ is adaptive then r simulates $\mathcal{A}$ step by step where each query is gathered in $2D$ rounds. This yields a round complexity of $O(D \cdot q)$, as desired. ◁

Application to minor-free graphs and Outerplanar graphs. For graphs which are minor-free with a bound $d = O(1)$ on the degree Newman and Sohler [42] argued the following in the centralized property testing model: (a) graph isomorphism is possible with constant number of queries, and (b) any property is testable with constant number of queries in the respective families of graphs. The latter means that there is an algorithm that performs a constant number of queries and succeeds with constant probability. Here, "constant" means independent of n.

For graphs which are forests and outerplanar (which include forests), Kusumoto and Yoshida [38] and Babu, Khoury, and Newman [3] argued, respectively, the following in the centralized property testing model: (a) graph isomorphism is possible with $\operatorname{poly}\log n$ number of queries, and (b) any property is testable with the same number of queries in the respective families of graphs. The latter means that there is an algorithm that performs $\operatorname{poly}\log n$ number of queries and succeeds with constant probability.

Their results are summarized in the following theorems.

▶ **Theorem 20** ([42, Thm. 3.2, 3.3]). *Given an oracle access to a minor-free graph with maximum degree $d = O(1)$, any graph property is testable with constant number of queries. This testing algorithm succeeds with constant probability. Specifically, testing isomorphism of two such graphs can be done in a constant number of queries and with constant probability.*

[13] We assume that the output of the algorithm is invariant to the labeling of the nodes and the port numbers of the edges. In the full version we remove this assumption and provide a general and more detailed proof.

▶ **Theorem 21** ([38, Thm. 1.3, 1.1], [3, Thm. 4.3, 4.2]). *Given an oracle access to a k-edge-outerplanar graph, any graph property is testable with* $\text{poly}\log n$ *queries.* [14] *This testing algorithm succeeds with constant probability. Specifically, testing isomorphism of two k-edge-outerplanar graph can be done in* $\text{poly}\log n$ *queries and with constant probability.*

Given the simulation argument above and Theorems 20, 21, we obtain the following corollaries.

▶ **Corollary 22.** *Any property is testable in the CONGEST model for minor-free graphs with maximum degree $d = O(1)$ within $O(D)$ rounds in the bounded-degree model. Specifically, this holds for the graph isomorphism problem.*

▶ **Corollary 23.** *Any property is testable in the CONGEST model for trees and k-edge-outerplanar graphs within $\tilde{O}(D)$ rounds in the general model. Specifically, this holds for the graph isomorphism problem.*

C Missing proofs and details

C.1 Proof of Claim 8

Proof. Let u and v be such that $|\triangle(N(u), N(v))| \geq \beta n$. The probability that C does not contain a node from $|\triangle(N(u), N(v))|$ is at most $(1-\beta)^s \leq \delta/n^2$, for an appropriate setting of s. Therefore, the claim follows by a union bound over all pairs u, v. ◀

C.2 Proof of Lemma 9

Proof. Let C be an ϵ-separating sequence of the nodes of G. Since π is an isomorphism between G are H, it follows that $\pi(C)$ is ϵ-separating in H (since for any u, v that have the same $\pi(C)$-label in H it must hold that $\pi^{-1}(u), \pi^{-1}(v)$ have the same C-label in G). Thus, by definition, for any pair of nodes v, u such that $\ell^H_{\pi(C)}(v) = \ell^H_{\pi(C)}(u)$ it holds that $|\triangle(N_H(u), N_H(v))| < \epsilon n$. Let f be $(C, \pi(C))$-label-consistent. By definition, $\ell^G_C(v) = \ell^H_{\pi(C)}(f(v))$, for every $v \in V(G)$. Therefore, there exists a bijection $g : V(H) \to V(H)$ such that $f = g \circ \pi$ and g only maps between nodes with the same $\pi(C)$-label. Thus, f can be obtained from π by making at most n swaps, one by one, between elements of $V(H)$ that have the same $\pi(C)$-label. Since each swap changes the adjacency matrix by at most ϵn, we obtain the desired result. ◀

C.3 Proof of Claim 10

Proof. Consider the outcome of querying the adjacency-matrices of G and H in $y \triangleq \Theta(\log(1/\delta)/\epsilon)$ random locations. Define the random variables $\{x_i\}_{i\in[y]}$ as follows: $x_i = 1$ if the values in the i-th location of both matrices are the same, and 0 otherwise. Let $p \triangleq \Delta(G, H)/n^2$ and define $\hat{p} = \sum_{i=1}^{y} x_i/|y|$. If $p = \epsilon/2$, then by the Multiplicative Chernoff's Bound, the probability that $\hat{p} > (3\epsilon)/2$ is at most δ (for the right setting of the parameter in the Θ-notation). Clearly, the same is true if $p < \epsilon/2$. On the other hand, if $p > \epsilon$ then by the Multiplicative Chernoff's Bound, the probability that $\hat{p} \leq (3\epsilon)/2$ is at most δ. Therefore, by accepting if and only if $\hat{p} \leq (3\epsilon)/2$ we can distinguish $\Delta(G, H) = p \cdot n^2 > \epsilon n^2$ from $\Delta(G, H) \leq \epsilon n^2/2$, as desired. ◀

[14] A graph G is 1-edge-outerplanar if it has a planar embedding in which all vertices of G are on the outer face. A graph G is k-edge-outerplanar if G has a planar embedding such that if all edges on the exterior face are deleted, the connected components of the remaining graph are all $(k-1)$-edge-outerplanar.

C.4 Selecting s Nodes u.a.r.

The procedure proceeds as follows. Each node selects a random number in $[n^{c+2}]$ where c is an absolute constant. For a fixed pair of nodes, the probability that both nodes pick the same number is at most $1/n^{c+2}$. Therefore, by union bound over all pairs, with probability at least $1 - 1/n^c$, all selected numbers are distinct. Conditioned on this event, the nodes with the s highest numbers are distributed uniformly at random. Each node sends its ID and its selected number up the BFS tree and the messages are forwarded up the tree in a manner that prioritizes messages whose number is higher. Therefore, the root receives the s highest numbers (along with the IDs of the corresponding nodes) in $D + s$ rounds. To see this observe that the message with the highest number is never delayed and in general the message with the i-highest number may be delayed for at most $i - 1$ rounds.

C.5 Matching of Y to Y'

We next describe the matching of Y to Y' according to g and explain how it is computed distributively. We aim to assign to each node in Y a label in $[|Y|]$ uniquely. This way each node in Y can match itself to a node in Y' (recall that all nodes know V_K and the total order on V_K). To this end, we use the BFS tree as follows. Each node in the BFS tree computes how many nodes in its subgraph are in Y. This can be done in $O(D)$ rounds as follows. We assume w.l.o.g. that each node knows its layer in the BFS tree. Let b denote the number of layers. We proceed in b rounds. In the first round, every node in Y which is in the b-th layer sends to its parent the message 1. In the next round, all the nodes in layer $b - 1$ sum up the messages they received and add 1 if they belong to Y. Then they send the result to their parents and so on until we end up at the root. Now the root partitions the interval $[1, \ldots, |Y|]$ into consecutive sub-intervals and assigns these sub-intervals to its children. Each child receives an interval whose size equals to the number of nodes in its subgraph that are in Y. In a similar manner, these sub-intervals are partitioned recursively down the tree until each node in Y is assigned with a unique number in $|Y|$, as desired.

Reaching a Consensus on Random Networks: The Power of Few

Linh Tran
Department of Mathematics, Yale University, New Haven, CT, USA
https://math.yale.edu/people/linh-tran
l.tran@yale.edu

Van Vu
Department of Mathematics, Yale University, New Haven, CT, USA
https://math.yale.edu/people/van-vu
van.vu@yale.edu

Abstract

A community of n individuals splits into two camps, Red and Blue. The individuals are connected by a social network, which influences their colors. Everyday, each person changes his/her color according to the majority of his/her neighbors. Red (Blue) wins if everyone in the community becomes Red (Blue) at some point.

We study this process when the underlying network is the random Erdos-Renyi graph $G(n,p)$. With a balanced initial state ($n/2$ persons in each camp), it is clear that each color wins with the same probability.

Our study reveals that for any constants p and ε, there is a constant c such that if one camp has $\frac{n}{2}+c$ individuals at the initial state, then it wins with probability at least $1-\varepsilon$. The surprising fact here is that c *does not* depend on n, the population of the community. When $p=1/2$ and $\varepsilon=.1$, one can set $c=6$, meaning one camp has $\frac{n}{2}+6$ members initially. In other words, it takes only 6 extra people to win an election with overwhelming odds. We also generalize the result to $p=p_n=\mathrm{o}(1)$ in a separate paper.

2012 ACM Subject Classification Mathematics of computing → Random graphs; Mathematics of computing → Graph theory; Mathematics of computing → Probability and statistics

Keywords and phrases Random Graphs Majority Dynamics Consensus

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.20

Category RANDOM

Related Version A full version of the paper is available at https://arxiv.org/abs/1911.10279.

Supplementary Material The simulation source code for the random process described in the paper is available at https://github.com/thbl2012/majority-dynamics-simulation

Acknowledgements We would like to thank A. Do, A, Ferber, A. Deneanu, J. Fox and X. Chen for inspiring discussions, H. V. Le, T. Can and L. T. D. Tran for proof-reading.

1 Introduction

1.1 The opinion exchange dynamics

Building mathematical models to explain how collective opinions are formed is an important and interesting task (see [12] for a survey on the topic, with examples from various fields, economy, sociology, statistical physics, to mention a few).

Obviously, our opinions are influenced by people around us, and this motivates the study of the following natural and simple model. A community of n individuals splits into two camps, Red and Blue, representing two competing opinions, which can be on any topic such


© Linh Tran and Van Vu;
licensed under Creative Commons License CC-BY

as brand competition, politics, ethical issues, etc. The individuals are connected by a social network, which influences their opinion on a daily basis (by some specific rule). We say that Red (respectively Blue) *wins* if everyone in the community becomes Red (respectively Blue) at some point.

We study this process when the underlying network is random. In this paper, we focus on the Erdos-Renyi random graph $G(n,p)$, which is the most popular model of random graphs [4, 10]. We use the majority rule, which is a natural choice. When a new day comes, a vertex scans its neighbors' colors in the previous day and adopts the dominant one. If there is a tie, it keeps its color.

▶ **Definition 1.** *The random graph $G(n,p)$ on $n \in \mathbb{N}$ vertices and density $p \in (0,1)$ is obtained by putting an edge between any two vertices with probability p, independently.*

1.2 Results

With a balanced initial state ($n/2$ persons in each camp), by symmetry, each color wins with the same probability $q < 1/2$, regardless of p. (Notice that there are graphs, such as the empty and complete graphs, on which no one wins.)

Our study reveals that for any given p and ε, there is a constant c such that if one camp has $\frac{n}{2} + c$ individuals at the initial state, then it wins with probability at least $1 - \varepsilon$. The surprising fact here is that c does not depend on n, the population of the community. When $p = 1/2$ and $\varepsilon = .1$, one can set c as small as 6.

▶ **Theorem 2** (The power of few)**.** *Consider the (majority) process on $G(n, 1/2)$. Assume that the Red camp has at least $\frac{n}{2} + 6$ vertices at the initial state, where $n \geq 300$. Then Red wins after the fourth day with probability at least* 90%*.*

This result can be stated without the Erd˝os-Renyi model; one can state an equivalent theorem by choosing the network uniformly, from the set of all graphs on n vertices.

This result reveals an interesting phenomenon, which we call "the power of few". The collective outcome can be extremely sensitive, as a modification of the smallest scale in the initial setting leads to the opposite outcome.

Our result applies in the following equivalent settings.

Model 1. We fix the two camps of size $n/2+6$ and $n/2-6$, respectively, and draw a random graph on their union.

Models 2. Draw a random graph first, let Red be a random subset of $n/2+6$ vertices (chosen uniformly from all subsets of that size), and Blue be the rest.

Model 3. Split the society into two camps of size $n/2$ each. Draw the random graph on their union, then recolor 6 random selected Blue vertices to Red.

Model 4. Split the society into two camps (Red and Blue) of size $n/2 - 6$ each and a "swinging" group (with no color yet) of 12 individuals. Draw the random graph on their union. Now let the swinging group join the Red camp.

With Model 3, we can imagine a balanced election process at the beginning. Then 6 persons change camp. This tiny group already guarantees the final win with an overwhelming odds. Similarly, Model 4 asserts that a swinging group of size 12 decides the outcome.

Our result can also be used to model the phenomenon that outcomes in seemingly identical situations become opposite. Consider two communities, each has exactly n individuals, sharing the same social network. In the first community, Red camp has size $n/2 + c$, and Blue camp has $n/2 - c$. In the second community, Blue camp has $n/2 + c$ and Red camp has $n/2 - c$. If

n is large, there is no way to tell the difference between the two communities. Even if we record everyone's initial opinion, clerical errors will surely swallow the tiny difference of $2c$. However, at the end, the collective opinion will be opposite, with high probability.

Now we state the general result for arbitrary constant density p.

▶ **Theorem 3** (Asymptotic bound). *Let p be a constant in $(0,1)$ and c_n be a positive integer which may depend on n . Assume that Red has $n/2 + c_n$ individuals in day zero and the random graph is $G(n,p)$. Then Red wins after the fourth day with probability at least $1 - K(p)\max\{n^{-1}, c_n^{-2}\}$, where $K(p)$ depends only on p.*

Both results follow from Theorem 6, which, in a slightly technical form, describes how the process evolves day by day. Our results can be extended to cover the case when there are more than 2 opinions; details will appear in a later paper [14].

1.3 Related results

Our problem is related to a well studied class of *opinion exchange dynamics* problems. In the field of Computer Science, loosely-related processes are studied in *population protocols* [2, 1], where individuals/agents/nodes choose their next state based on that of their neighbors. The most separating difference is the network, as connections in these models often randomly change with time, while our study concerns a fixed network (randomly generated before the process begins).

The survey by Mossel and Tamuz [12] discussed severals models for these problems, including the *DeGroot model* [6], where an individual's next state is a weighted average of its neighbors' current states, the *voter model* [5], where individuals change states by emulating a random neighbor each day. The *majority dynamics* model is in fact the same as ours, and is also more popular than the other two, having been studied in [11, 8, 3]. The key difference, as compared to our study, is in the set-ups. In these earlier papers, each individual chooses his/her initial color uniformly at random. The central limit theorem thus guarantees that with high probability, the initial difference between the two camps is of order $\Theta(\sqrt{n})$. Therefore, these papers did not touch upon the "power of few" phenomenon, which is our key message. On the other hand, they considered sparse random graphs where the density $p = p_n$ goes to zero as $n \to +\infty$.

In [3], Benjamini, Chan, O'Donnell, Tamuz, and Tan considered random graphs with $p \geq \lambda n^{-1/2}$, where λ is a sufficiently large constant, and showed that the dominating color wins with probability at least .4 [3, Theorem 1.2], while conjecturing that this probability in fact tends to 1 as $n \to \infty$. This conjecture was proved by Fountoulakis, Kang, and Makai [8, Theorem 1.1].

▶ **Theorem 4.** *For any $0 < \varepsilon \leq 1$ there is $\lambda = \lambda(\varepsilon)$ such that the following holds for $p \geq \lambda n^{-1/2}$. With probability at least $1 - \varepsilon$, over the choice of the random graph $G(n,p)$ and the choice of the initial state, the dominating color wins after four days.*

For related results on random regular graphs, see [11, 12].

1.4 Extension for sparse random graphs

Note that the results presented in this paper only applies for a constant p, which, in the context of $G(n,p)$, produces *dense graphs*. For *sparse graphs*, i.e. when $p = p_n$ depends on n and tends to 0 as $n \to +\infty$, the main ideas in this paper can be used, but with slightly different algebraic techniques, to obtain a similar result.

▶ **Theorem 5.** *For any $0 < \varepsilon \leq 1$ there is $c = c(\varepsilon)$ such that the following holds for $p \geq (2 + o(1))(\log n)/n$. Assume that Red camp has size at least $n/2 + c/p$ initially, then it wins with probability at least $1 - \varepsilon$.*

The technical changes needed to prove this theorem require rewriting entire proofs with new computations, so we leave the proof to our future paper [14]. Additional information such as the length of the process and the explicit relation between the bound with p and c will also be discussed there. Notice that when p is a constant, this result covers the "Power of Few" phenomenon as a special case, albeit with c much larger than 6. Therefore, the techniques and results in this paper still have merit since they achieve a specific, surprisingly small constant. Theorem 5 no longer holds for $p < (\log n)/n$ as in this case there are, with high probability, isolated vertices. Any of these vertices keeps it original color forever. In this case, the number of Blue vertices converges with time, and we obtain a bound on the limit in [14].

One can use Theorem 5 to derive a "delayed" version (in which Red may need more than 4 days) of Theorem 4, by first proving that with high probability, one side gains an advantage of size at least $C\sqrt{n}$ after the first day, for some constant C. This "majority side" then wins with high probability given $p \geq \lambda n^{-1/2}$ (which satisfies the requirement $p \geq (2 + o(1))(\log n)/n$) with λ sufficiently large so that $\lambda C = pC\sqrt{n}$ is large. The detailed argument is in Appendix A.3.

1.5 Notation

- R_t, B_t: Respectively the sets of Red and Blue vertices after day t. (At this point each person has updated their color t times.)
- $\mathrm{I}_t(u) \stackrel{\text{def}}{=} \mathbf{1}_{\{u \in R_t\}}$: $\{0,1\}$-indicator of the event that u is Red after day t.
- $\mathrm{J}_t(u) \stackrel{\text{def}}{=} 2\mathrm{I}_t(u) - 1$: $\{-1,1\}$-indicator of the same event.
- $u \sim v \equiv (u,v) \in E$: Event that u and v are adjacent.
- $\Gamma(v) \stackrel{\text{def}}{=} \{u : u \sim v\}$: The neighborhood of v.
- $W_{uv} \stackrel{\text{def}}{=} \mathbf{1}_{\{u \sim v\}}$ - Indicator of the adjacency between u and v.
- $\mathcal{N}(\mu, \sigma^2)$: The Normal Distribution with mean μ and variance σ^2.
- $\Phi(a,b) \stackrel{\text{def}}{=} (2\pi)^{-1/2} \int_a^b e^{-\frac{x^2}{2}} dx$ and $\Phi(a) \stackrel{\text{def}}{=} \Phi(-\infty, a)$, $\Phi_0(a) \stackrel{\text{def}}{=} \Phi(0,a)$.

1.6 Main Theorem

The main theorem concerns dense graphs, where p is at least a constant. When given appropriate values for the parameters, it implies the "Power of Few" phenomenon in Theorem 2. Before stating the theorem, we define some expressions.

$$C_0 \stackrel{\text{def}}{=} 0.56, \quad C_1 \stackrel{\text{def}}{=} \sqrt{3\log 2}, \quad \sigma = \sigma(p) \stackrel{\text{def}}{=} \sqrt{p(1-p)},$$

$$P_1 = P_1(n,p,c,\varepsilon_2) \stackrel{\text{def}}{=} \frac{1/4 + 4C_0^2\left(1-2\sigma^2\right)^2 \cdot \frac{n-1}{n-2}}{\left(\sqrt{n-1}\Phi_0\left(\frac{2pc+\min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) - C_0\frac{1-2\sigma^2}{\sigma} - \frac{C_1+\varepsilon_2}{2p} - \frac{1}{2\sqrt{n}}\right)^2},$$

$$P_2 = P_2(n,\varepsilon_2,\varepsilon_1) \stackrel{\text{def}}{=} \frac{1}{n}\exp\left(-n\left[(1-2\varepsilon_1)\varepsilon_2 - \varepsilon_1\right]\right),$$

$$P_3 = P_3(n,p,\varepsilon_1) \stackrel{\text{def}}{=} \frac{1}{n}\exp\left(-\tfrac{1}{2}p^3(2\varepsilon_1 n - 1)^2 + 2n\log 2\right),$$

$$P_4 = P_4(n,p) \stackrel{\text{def}}{=} n\exp\left(-\tfrac{2}{9}p^2(n-1)\right).$$

▶ **Theorem 6.** *Let $p \in (0,1)$, $c \in \mathbb{N}$, $n \in \mathbb{N}$, and $\varepsilon_1, \varepsilon_2 > 0$. Define C_0, C_1, σ, P_1, P_2, P_3, P_4 as above. Assume that*

$$2\sqrt{n-1}\Phi_0\left(\frac{2pc + \min\{p, 1-p\}}{\sigma\sqrt{n-1}}\right) > \frac{C_1 + \varepsilon_2}{p} + \frac{1}{\sqrt{n}} \quad \text{and} \quad 2\varepsilon_1 n > 1. \tag{1}$$

With n^R, n^B being integers such that $n^R + n^B = n$ and $1 \le n^B \le \frac{n}{2} - c$, the election process on $G \sim G(n,p)$ with $|B_0| = n^B$ satisfies the following

1. *With $n_0^B = \frac{n}{2} - c$, $n_1^B = \frac{n-1}{2} - \left(\frac{C_1+\varepsilon_2}{2p}\right)\sqrt{n}$, $n_2^B = \left(\frac{1}{2} - \varepsilon_1\right)n$, $n_3^B = \frac{1}{3}p(n-1)$, $n_4^B = 0$, we have $\mathbf{P}\left(|B_t| \le n_t^B \mid |B_{t-1}| \le n_{t-1}^B\right) \ge 1 - P_t$ for each $t = 1, 2, 3, 4$.*
2. *$\mathbf{P}\left(R_4 = V(G) \mid |B_0| = n^B\right) \ge 1 - (P_1 + P_2 + P_3 + P_4)$.*

Intuitively, P_i is a upper bound on the probability of some abnormal event happening on Day i. If none of these "catastrophes" occur, the whole population becomes Red after Day Four. Note that if one let $n \to +\infty$ while fixing all other parameters, P_2, P_3 and P_4 all tend to 0, leaving P_1 as the main asymptotic component of the probability bound.

The proof for this theorem has two main parts corresponding to the next two sections. In Section 2, we apply a concentration bound to the number of Red vertices after Day 1 to show that with probability at least $1 - P_1$, this number is at least $n - n_1^B$. In Section 3, we show that with high probability, this $\Omega(n^{1/2})$ advantage leads Red to win, using a *shrinking argument* that bypasses the dependency of the coloring on the current day.

From Theorem 6, one can deduce Theorems 2 and 3 in a few steps. Detailed proofs appear in Appendix A.1.

1.7 Open questions

Let $\rho(k,n)$ be the probability that Red win if its camp has size $n/2 + k$ in the beginning, when $p = .5$. Theorem 2 shows that $\rho(6,n) \ge .9$ (given that n is sufficiently large). In other words, six defectors guarantee Red's victory with an overwhelming odd. In fact, we have $\rho(4,n) \ge .7$ by plugging in the same values for ε_1 and ε_2 with $c = 4$ in Theorem 2's proof. We conjecture that one defector already brings a non-trivial advantage.

▶ **Conjecture 7** (The power of one)**.** *There is a constant $\delta > 0$ such that $\rho(1,n) \ge 1/2 + \delta$ for all sufficiently large n.*

In the following numerical experiment, we run $T = 10000$ independent trials. In each trial, we fix a set of $N = 10000$ nodes with 5001 Red and 4999 Blue (meaning $c = 1$), generate a graph from $G(N, 1/2)$, and simulate the process on the resulting graph. We record the number of wins and the number of days to achieve the win in percentage in Table 1. Among others, we see that Red wins within 3 days with frequency more than .9. The source code for the simulation along with execution instructions can be found online at `https://github.com/thbl2012/majority-dynamics-simulation`.

Imagine that people defect from Blue camp to Red camp one by one. The *value* of the ith defector is defined as $v(i,n) = \rho(i,n) - \rho(i-1,n)$ (where we take $\rho(n,0) = 1/2$). It is intuitive to think that the values of the defectors decrease. (Clearly defector number $n/2$ adds no value.)

▶ **Conjecture 8** (Values of defectors)**.** *For any fixed i and sufficiently large n, we have $v(i,n) \ge v(i+1,n)$.*

It is clear that the Conjecture 8 implies Conjecture 7, with $\delta = \frac{4}{5} = .08$, although the simulation results above suggests that δ can be at least .43.

Table 1 Winners and winning days with their frequencies.

T	p	Red	Blue	Winner	Last day	Count	Frequency
10^4	1/2	5001	4999	Blue	3	496	4.96 %
10^4	1/2	5001	4999	Blue	4	77	0.77 %
10^4	1/2	5001	4999	Blue	5	3	0.03 %
10^4	1/2	5001	4999	Blue	7	1	0.01 %
10^4	1/2	5001	4999	Red	2	25	0.25 %
10^4	1/2	5001	4999	Red	3	9313	93.13 %
10^4	1/2	5001	4999	Red	4	85	0.85 %

2 Day One

At day one, the number of Red and Blue neighbors of each node v are both binomial random variables, with means roughly $n/2 + c$ and $n/2 - c$ respectively. The central limit theorem then implies that most of their masses are concentrated within an interval of length $\Theta(\sqrt{n})$ around their respective expectations. A subinterval of constant length in this interval will have $\Theta(n^{-1/2})$ mass. Therefore, one expects that the probability that the number of Red exceeds the number of Blues (in that particular neighborhood) is $1/2 + \Omega(n^{-1/2})$. Thus, the expectation of Red nodes after the first day is $n/2 + \Omega(n^{1/2})$. We consolidate this intuition in the main result of this section, Theorem 9.

Firstly, let us recall a few terms defined in Section 1.6.

$$\sigma \stackrel{\text{def}}{=} \sqrt{p(1-p)}, \quad \text{and } P_1 \stackrel{\text{def}}{=} \frac{\frac{1}{4} + 4C_0^2\left(1-2\sigma^2\right)^2 \cdot \frac{n-1}{n-2}}{\left(\sqrt{n-1}\Phi_0\left(\frac{2pc+\min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) - \frac{C_0(1-2\sigma^2)}{\sigma} - \frac{C_1+\varepsilon_2}{2p} - \frac{1}{2\sqrt{n}}\right)^2}.$$

Define a new term Q by

$$Q = Q(n,p,c,d) \stackrel{\text{def}}{=} \frac{\frac{1}{4} + 4C_0^2\left(1-2\sigma^2\right)^2 \cdot \frac{n-1}{n-2}}{\left(\sqrt{n-1}\Phi_0\left(\frac{2pc+\min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) - \frac{C_0(1-2\sigma^2)}{\sigma} - d - \frac{1}{2\sqrt{n}}\right)^2}.$$

Observe that $Q\left(n,p,c,\frac{C_1+\varepsilon_2}{2p}\right) = P_1(n,p,c,\varepsilon_2)$. The following result thus covers the first day in Theorem 6 by just plugging in $d = (C_1+\varepsilon_2)/(2p)$.

▶ **Theorem 9.** *Let $p \in (0,1)$ and c be constants and σ and Q be defined above. Then if $n, n^R, n^B \in \mathbb{N}$ such that $n^R + n^B = n$, $1 \le n^B \le \frac{n}{2} - c$. Then for all $d \in \mathbb{R}_{>0}$ and $n \in \mathbb{N}$ such that*

$$\sqrt{n-1}\Phi_0\left(\frac{2pc+\min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) - \frac{C_0(1-2\sigma^2)}{\sigma} > d + \frac{1}{2\sqrt{n}}, \tag{2}$$

we have

$$\mathbf{P}\left(|B_1| > \frac{n-1}{2} - d\sqrt{n} \;\middle|\; |B_0| = n^B\right) \le Q\left(n,p,c,d\right).$$

The crux of the proof relies on some preliminary results regarding the difference of two binomial random variables, which we discuss next.

2.1 Background on difference of Binomial Random Variables

The difference of two binomial random variables with the same probability p be written as a sum of independent random variables, each of which is either a $\mathrm{Bin}(1,p)$ variable or minus of one. A natural way to bound this sum is done via a Berry-Esseen normal approximation.

▶ **Theorem 10** (Berry-Esseen). *Let n be any positive integer. If $X_1, X_2, X_3, \dots, X_n$ are random variables with zero means, variances $\sigma_1^2, \sigma_2^2, \cdots, \sigma_n^2 > 0$, and absolute third moments $\mathbf{E}\left[|X_i|^3\right] = \rho_i < \infty$, we have:*

$$\sup_{x\in\mathbb{R}} \left| \mathbf{P}\left(\sum_{i=1}^{n} X_i \le x\right) - \Phi\left(\frac{x}{\sigma_X}\right)\right| \le C_0 \cdot \frac{\sum_{i=1}^n \rho_i}{\sigma_X^3},$$

where $\sigma_X = \left(\sum_{i=1}^n \sigma_i^2\right)^{1/2}$ and $C_0 = .56$ is a constant.

The original proof by Esseen [7] yielded $C_0 = 7.59$, and this constant has been improved a number of times. The latest work by Shevtsova [13] achieved $C_0 = .56$, which will be used for the rest of the paper. A direct application of this theorem gives the following lemma.

▶ **Lemma 11.** *For $p \in (0,1)$, $\sigma = \sqrt{p(1-p)}$ and $n_1, n_2 \in \mathbb{N}$ such that $n_1 > n_2$. let $Y_1 \sim Bin(n_1, p)$, $Y_2 \sim Bin(n_2, p)$ be independent random variables. Then for any $d \in \mathbb{R}$,*

$$\mathbf{P}\left(Y_1 > Y_2 + d\right) \ge \frac{1}{2} + \Phi_0\left(\frac{p(n_1-n_2)-d}{\sigma\sqrt{n_1+n_2}}\right) - \frac{C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n_1+n_2}}.$$

Proof. By definition, the difference $Y = Y_1 - Y_2$ can be expressed as

$$X = X_1 + X_2 + X_3 + \cdots + X_{n_1+n_2},$$

where all X_i's are independent and either $X_i \sim \mathrm{Bin}(1,p)$ or $-X_i \sim \mathrm{Bin}(1,p)$. Then $\mathbf{E}[X] = \sum_i \mathbf{E}[X_i] = p(n_1-n_2)$. For all i, $\mathbf{Var}[X_i] = \sigma^2$ and $\mathbf{E}\left[|X_i - \mathbf{E}[X_i]|^3\right] = p(1-p)^3 + (1-p)p^3 = \sigma^2(1-2\sigma^2)$ Applying Theorem 10, we have

$$\begin{aligned}
\mathbf{P}\left(Y_1 \le Y_2 + d\right) &= \mathbf{P}\left(X - \mathbf{E}[X] \le d - p(n_1-n_2)\right) \\
&\le \Phi\left(\frac{d-p(n_1-n_2)}{\sigma_X}\right) + C_0\frac{\sum_i \mathbf{E}\left[|X_i - \mathbf{E}[X_i]|^3\right]}{\sigma_X^3} \\
&= \Phi\left(\frac{d-p(n_1-n_2)}{\sigma\sqrt{n_1+n_2}}\right) + C_0\frac{\sigma^2(1-2\sigma^2)(n_1+n_2)}{\sigma^3(n_1+n_2)^{3/2}} \\
&= \frac{1}{2} - \Phi_0\left(\frac{p(n_1-n_2)-d}{\sigma\sqrt{n_1+n_2}}\right) + \frac{C_0(1-2\sigma^2)}{\sigma\sqrt{n_1+n_2}},
\end{aligned}$$

and the claim follows by taking the complement event. ◀

▶ **Lemma 12.** *Let $p \in (0,1)$ be a constant and $\sigma = \sqrt{p(1-p)}$, $X_1 \sim Bin(n_1,p)$ and $X_2 \sim Bin(n_2,p)$ be independent r.v.s. Then for any integer d,*

$$\mathbf{P}\left(X_1 = X_2 + d\right) \le \frac{2C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n_1+n_2}}.$$

Proof. Let $n = n_1 + n_2$ and $\mu = \mathbf{E}[X_1] - \mathbf{E}[X_2] = p(n_1-n_2)$. Fix $\varepsilon \in (0,1)$, by the same computations in Lemma 11, we have

$$\mathbf{P}\left(X_1 - X_2 \le d - \varepsilon\right) \ge \Phi\left(\frac{d-\mu-\varepsilon}{\sigma\sqrt{n}}\right) - \frac{C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n}},$$

$$\mathbf{P}\left(X_1 - X_2 < d + \varepsilon\right) \le \Phi\left(\frac{d-\mu+\varepsilon}{\sigma\sqrt{n}}\right) + \frac{C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n}}.$$

It follows that

$$\begin{aligned}\mathbf{P}(X_1 = X_2 + d) \;&\le\; \mathbf{P}(d-\varepsilon < X_1 - X_2 < d+\varepsilon)\\ &\le\; \Phi\left(\frac{d-\mu+\varepsilon}{\sigma\sqrt{n}}\right) - \Phi\left(\frac{d-\mu-\varepsilon}{\sigma\sqrt{n}}\right) + \frac{2C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n}}.\end{aligned}$$

Letting $\varepsilon \to 0$, we obtain the desired claim. ◀

Proof of Theorem 9

Recall that $\left|R_1\right| = n - \left|B_1\right|$. Our goal is to lower-bound the probability that $\left|R_1\right| < \frac{n+1}{2} + d\sqrt{n}$ for any given constant d. Recall the indicator $\mathrm{I}_1(v)$ which is 1 if v is Red after Day One and 0 otherwise. We have:

$$\left|R_1\right| = \sum_{v\in V} \mathrm{I}_1(v).$$

Since the indicators are not independent, a natural choice for bounding their sum is to use Chebysev's inequality. We proceed in two steps:

1. Lower-bound $\mathbf{E}\left[\left|R_1\right|\right]$ by lower-bounding each term $\mathbf{E}\left[\mathrm{I}_1(v)\right]$.
2. Upper-bound $\mathbf{Var}\left[\left|R_1\right|\right]$ by upper-bounding each $\mathbf{Var}\left[\mathrm{I}_1(v)\right]$ and $\mathbf{Cov}\left[\mathrm{I}_1(v), \mathrm{I}_1(v')\right]$.

▷ Claim 13. $\mathbf{E}\left[\left|R_1\right|\right] \;\ge\; \dfrac{n+1}{2} + \left(T(n,p,c,d) + d\right)\sqrt{n}$,

where $T(n,p,c) \stackrel{\text{def}}{=} \sqrt{n-1}\,\Phi_0\left(\dfrac{2pc + \min\{p, 1-p\}}{\sigma\sqrt{n-1}}\right) - \dfrac{C_0\left(1-2\sigma^2\right)}{\sigma}$.

Proof. For each vertex v, let $d_0^R(v)$ and $d_0^B(v)$ respectively be the numbers of its Red and Blue neighbors before the first day. By our rule, the event $\{v \in R_1\}$ is equivalent to $d_0^R(v) > d_0^B(v)$ if $v \in B_0$, i.e. $\mathrm{I}_0(v) = 0$, and to $d_0^R(v) \ge d_0^B(v)$ if $v \in R_0$, i.e. $\mathrm{I}_0(v) = 1$. This implies

$$\forall v \in V.\, \left[v \in R_1 \;\Leftrightarrow\; d_0^R(v) + \mathrm{I}_0(v) > d_0^B(v)\right]. \tag{3}$$

Note that $d_0^R(v) \sim \mathrm{Bin}\left(n^R - \mathrm{I}_0(v), p\right)$ and $d_0^B(v) \sim \mathrm{Bin}\left(n^B + \mathrm{I}_0(v) - 1, p\right)$. By Lemma 11, we have:

$$\begin{aligned}&\mathbf{E}\left[\mathrm{I}_1(v)\right] = \mathbf{P}\left(v \in R_1\right) = \mathbf{P}\left(d_0^R(v_i) + \mathrm{I}_0(v_1) > d_0^B(v_i)\right)\\ &\ge \frac{1}{2} + \Phi_0\left(\frac{p\left(n^R - n^B + 1 - 2\mathrm{I}_0(v)\right) + \mathrm{I}_0(v)}{\sigma\sqrt{n^R + n^B - 1}}\right) - \frac{C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n^R + n^B - 1}}\\ &= \frac{1}{2} + \Phi_0\left(\frac{2pc + p_v}{\sigma\sqrt{n-1}}\right) - \frac{C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n-1}},\end{aligned}$$

where $p_v \stackrel{\text{def}}{=} p\left(1 - \mathrm{I}_0(v)\right) + (1-p)\mathrm{I}_0(v) \ge \min\{p, 1-p\}$. Now

$$\begin{aligned}&\mathbf{E}\left[\left|R_1\right|\right] = \sum_{v\in V}\mathbf{E}\left[\mathrm{I}_1(v)\right] \ge \sum_{v\in V}\Phi_0\left(\frac{2pc + \min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) + n\left(\frac{1}{2} - \frac{C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n-1}}\right)\\ &\ge \frac{n}{2} + \left[\sqrt{n-1}\,\Phi_0\left(\frac{2pc + \min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) - \frac{C_0\left(1-2\sigma^2\right)}{\sigma}\right]\sqrt{n}\\ &= \frac{n}{2} + \left(T(n,p,c,d) + d + \frac{1}{2\sqrt{n}}\right)\sqrt{n} = \frac{n+1}{2} + \left(T(n,p,c,d) + d\right)\sqrt{n},\end{aligned}$$

The proof is complete. ◁

▷ Claim 14. $\mathbf{Var}\left[\left|R_1\right|\right] \leq \frac{n}{4} + 4C_0^2\left(1-2\sigma^2\right)^2 \cdot \frac{n(n-1)}{n-2}$.

Proof. We first have: $\mathbf{Var}\left[\left|R_1\right|\right] = \sum_{i=1}^{k} \mathbf{Var}\left[\mathrm{I}_1(v_1)\right] + 2\sum_{v_1 \neq v_2} \mathbf{Cov}\left[\mathrm{I}_1(v_1), \mathrm{I}_1(v_2)\right]$.
The variances $\mathbf{Var}\left[\mathrm{I}_1(v)\right]$ are easy due to $\mathrm{I}_1(v)$ being a Bernoulli r.v.:

$$\mathbf{Var}\left[\mathrm{I}_1(v)\right] = \mathbf{E}\left[\mathrm{I}_1(v)\right]\left(1-\mathbf{E}\left[\mathrm{I}_1(v)\right]\right) = \frac{1}{4} - \left(\mathbf{E}\left[\mathrm{I}_1(v)\right] - \frac{1}{2}\right)^2 \leq \frac{1}{4}. \tag{4}$$

Bounding the covariance $\mathbf{Cov}\left[\mathrm{I}_1(v_1), \mathrm{I}_1(v_2)\right]$ for two distinct vertices v_1, v_2 requires a bit more care, as the indicators are not independent. By definition

$$\mathbf{Cov}\left[\mathrm{I}_1(v_1), \mathrm{I}_1(v_2)\right] = \mathbf{P}\left(v_1, v_2 \in R_1\right) - \mathbf{P}\left(v_1 \in R_1\right)\mathbf{P}\left(v_2 \in R_1\right).$$

Consider the event $\{v_1, v_2 \in R_1\}$; $\mathbf{P}\left(v_1, v_2 \in R_1\right)$ can be written as

$$\mathbf{P}\left(v_1, v_2 \in R_1 | v_1 \sim v_2\right)\mathbf{P}(v_1 \sim v_2) + \mathbf{P}\left(v_1, v_2 \in R_1 | v_1 \not\sim v_2\right)\mathbf{P}(v_1 \not\sim v_2).$$

Notice that after we specify the adjacency between v_1 and v_2, the remaining vertices in the neighborhoods of v_1 and v_2 are independent. Letting $a_i = \mathbf{P}(v_i \in R_1 \mid v_1 \sim v_2)$, $b_i = \mathbf{P}(v_i \in R_1 \mid v_1 \not\sim v_2)$ and using shorthand $q \stackrel{\text{def}}{=} 1-p$, we have

$$\mathbf{P}\left(v_1, v_2 \in R_1\right) = pa_1a_2 + (1-p)b_1b_2.$$

Now consider $\mathbf{P}\left(v_1 \in R_1\right)\mathbf{P}\left(v_2 \in R_1\right)$. Splitting up the two events by $\{v_1 \sim v_2\}$ gives $\mathbf{P}\left(v_1 \in R_1\right) = pa_1 + qb_1$ and $\mathbf{P}\left(v_2 \in R_1\right) = pa_2 + qb_2$. Putting everything together, we have

$$\begin{aligned}\mathbf{Cov}\left[\mathrm{I}_1(v_1), \mathrm{I}_1(v_2)\right] &= pa_1a_2 + qb_2b_2 - \left(pa_1 + qb_1\right)\left(pa_2 + qb_2\right) \\ &= pq(a_1-b_1)(a_2-b_2) = \sigma^2(a_1-b_1)(a_2-b_2).\end{aligned} \tag{5}$$

We next analyze the relationship between a_1 and b_1. (The analysis for a_2 and b_2 is similar). Define

$$\begin{aligned}m^R &\stackrel{\text{def}}{=} \left|R_0 \setminus \{v_1, v_2\}\right| &&= n^R - \left(\mathrm{I}_0(v_1) + \mathrm{I}_0(v_2)\right), \\ m^B &\stackrel{\text{def}}{=} \left|B_0 \setminus \{v_1, v_2\}\right| &&= n^B + \left(\mathrm{I}_0(v_1) + \mathrm{I}_0(v_2)\right) - 2, \\ d^R &\stackrel{\text{def}}{=} \left|\left(\Gamma(v_1) \cap R_0\right) \setminus \{v_2\}\right| &&= d_0^R(v_1) - \mathrm{I}_0(v_2)W_{v_1v_2} \\ d^B &\stackrel{\text{def}}{=} \left|\left(\Gamma(v_1) \cap B_0\right) \setminus \{v_2\}\right| &&= d_0^B(v_1) + \left(\mathrm{I}_0(v_2) - 1\right)W_{v_1v_2}\end{aligned}$$

We have $m^R + m^B = n-2$, $d^R \sim \mathrm{Bin}(m^R, p)$, $d^B \sim \mathrm{Bin}(m^B, p)$ and $d_0^R(v_1) - d_0^B(v_1) = d^R - d^B + \mathrm{J}_0(v_2)W_{v_1v_2}$. Now we can rewrite a_1 and b_1 using (3) in terms of the above:

$$\begin{aligned}a_1 &= \mathbf{P}\left(d_0^R(v_1) + \mathrm{I}_0(v_1) > d_0^B(v_1) \;\middle|\; v_1 \sim v_2\right) = \mathbf{P}\left(d^R - d^B > -\mathrm{J}_0(v_2) - \mathrm{I}_0(v_1)\right), \\ b_1 &= \mathbf{P}\left(d_0^R(v_1) + \mathrm{I}_0(v_1) > d_0^B(v_1) \;\middle|\; v_1 \not\sim v_2\right) = \mathbf{P}\left(d^R - d^B > -\mathrm{I}_0(v_1)\right).\end{aligned}$$

Case analysis on $\mathrm{J}_0(v_2)$: $\begin{cases}\mathrm{J}_0(v_2) = -1 & \implies a_1 - b_1 = \mathbf{P}\left(d^R - d^B = -\mathrm{I}_0(v_1)\right) \\ \mathrm{J}_0(v_2) = 1 & \implies b_1 - a_1 = \mathbf{P}\left(d^R - d^B = -1 - \mathrm{I}_0(v_1)\right).\end{cases}$

In any case, by Lemma 12 we have $\left|a_1 - b_1\right| \leq \frac{2C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{m^R + m^B}} = \frac{2C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n-2}}$.
The same analysis for a_2 and b_2, and Equation (5) then imply

$$\mathbf{Cov}\left[\mathrm{I}_1(v_1), \mathrm{I}_1(v_2)\right] \leq \sigma^2 \cdot \frac{2C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n-2}} \cdot \frac{2C_0\left(1-2\sigma^2\right)}{\sigma\sqrt{n-2}} = \frac{4C_0^2\left(1-2\sigma^2\right)^2}{n-2}. \tag{6}$$

Equations (4) and (6) together yield

$$\mathbf{Var}\left[\left|R_1\right|\right] \le \frac{1}{4}\cdot n + 2\cdot\frac{4C_0^2\left(1-2\sigma^2\right)^2}{n-2}\cdot\binom{n}{2} = \frac{n}{4} + 4C_0^2\left(1-2\sigma^2\right)^2\cdot\frac{n(n-1)}{n-2}. \tag{7}$$

The proof is complete. ◁

From Claims 13 and 14, a standard Chebyshev's inequality gives

$$\begin{aligned}\mathbf{P}\Big(\left|B_1\right| > \tfrac{n-1}{2} - d\sqrt{n}\Big) &= \mathbf{P}\Big(\left|R_1\right| < \tfrac{n+1}{2} + d\sqrt{n}\Big) \le \frac{\mathbf{Var}\left[\left|R_1\right|\right]}{\left(\mathbf{E}\left[\left|R_1\right|\right] - \frac{n+1}{2} - d\sqrt{n}\right)^2} \\ &\le \frac{\frac{n}{4} + 4C_0^2\left(1-2\sigma^2\right)^2\cdot\frac{n(n-1)}{n-2}}{T(n,p,c,d)^2 n} = Q(n,p,c,d).\end{aligned}$$

The proof of Theorem 9 is complete. This theorem forms the first part of Theorem 6, which shrinks the Blue camp from size $\frac{n}{2}-c$ to $\frac{n}{2}-\Omega(\sqrt{n})$. We state explicitly the relevant result to wrap up this section.

▶ **Corollary 15.** *For any $p\in(0,1)$, $c>0$, $\varepsilon_2>0$ and $n\in\mathbb{N}$, if the Blue side starts with at most $\frac{n}{2}-c=n_0^B$ members, it shrinks to size at most $\frac{n-1}{2}-\left(\frac{C_1+\varepsilon_2}{2p}\right)\sqrt{n}=n_1^B$ with probability at least $1-P_1(n,p,c,\varepsilon_2)$.*

3 Day Two and after

Next, we analyze the situation after the first day. Clearly, if one fixes the coloring after Day 1 and examine the graph, its distribution is no longer $G(n,p)$. Therefore, we cannot apply the same method in proving Theorem 9 for later days. Instead, we use "shrinking arguments" to argue that it is likely for the Blue camp to monotonously shrink to empty, regardless of the choice of its members, due to G's structure.

The core of our shrinking argument is *Hoeffding's inequality*, a classical result that gives exponentially small probability tails for sums of independent random variables.

▶ **Theorem 16** (Hoeffding's inequality). *Let $\{X_i\}_{i=1}^n$ be independent random variables and $\{a_i\}_{i=1}^n$, $\{b_i\}_{i=1}^n$, such that for all $i=1,2,\cdots,n$, $a_i\le X_i\le b_i$ almost surely. Then for $X=X_1+X_2+\cdots+X_n$, we have*

$$\max\Big\{\mathbf{P}\left(X-\mathbf{E}\left[X\right]\ge t\right),\ \mathbf{P}\left(X-\mathbf{E}\left[X\right]\le -t\right)\Big\} \le \exp\left(-\frac{2t^2}{\sum_{i=1}^n(b_i-a_i)^2}\right).$$

The proof of Hoeffding's inequality is available in most graduate level probability textbooks, e.g. [15]. The original proof given by Hoeffding appeared in [9].

A simple yet useful shrinking argument is that, in the $G(n,p)$ model, it is with high probability that all vertices in G have many neighbors, so a small enough Blue camp will not be able to influence anyone by a majority, thus inevitably vanishes the next day.

▶ **Lemma 17.** *For $p\in(0,1)$ and $n\in\mathbb{N}_{>1}$, with probability at least*

$$1-n\exp\left(-\tfrac{2}{9}p^2(n-1)\right) = 1-P_4(n,p),$$

G is such that all vertices have more than $\frac{2}{3}p(n-1)$ neighbors, thus any choice of the Blue camp of at most $\frac{1}{3}p(n-1)=n_3^B$ members shrinks to $0=n_4^B$ the next day.

The proof is standard using the Hoeffding bound, so we refer to Appendix A.2 for details.

This simple lemma forms the first block of our shrinking argument. The overall aim is to argue that with high probability, G is such that a Blue camp of size $\frac{n}{2} - \mathrm{O}(\sqrt{n})$ in Day 1 will inevitably be reduced to size $\mathrm{O}(n)$ in Day 2, and then to size less than $\frac{1}{2}p(n-1)$ in Day 3, then vanishes in the fourth day by Lemma 17. The remaining blocks, which correspond to days before the fourth, will require a more complicated argument.

▶ **Lemma 18.** *Let $p \in (0,1)$, $n, n_0 \in \mathbb{N}$, $n_0 < \frac{n}{2}$. Then for all $m \in \mathbb{N}, m \leq n$, with probability at least*

$$1 - \frac{4^n}{n} \exp\left(-\frac{2p^2(n-2n_0-1)^2 m}{n+m-2}\right),$$

G is such that any choice of the Blue camp of at most n_0 members shrinks to below m in the next day.

Proof. Consider a subset S of V with m elements. We will first bound the probability that S entirely turn Blue the next day. Let (R, B) be the initial coloring with $\big|B\big| = n_0 < n - n_0 = \big|R\big|$. For each $v \in V$, let $\mathrm{dif}(v) \stackrel{\text{def}}{=} \big|\Gamma(v) \cap R\big| - \big|\Gamma(v) \cap B\big|$, and let $\mathrm{dif}(S) \stackrel{\text{def}}{=} \sum_{v \in S} \mathrm{dif}(v)$. We break down each $\mathrm{dif}(v)$ and $\mathrm{dif}(S)$ as follows:

$$\begin{aligned} \mathrm{dif}(v) &= \sum_{u \in R \cap S} W_{vu} + \sum_{u \in R \setminus S} W_{vu} - \sum_{u \in B \cap S} W_{vu} - \sum_{u \in B \setminus S} W_{vu} \\ \mathrm{dif}(S) &= \sum_{v \in S} \sum_{u \in R \cap S} W_{vu} + \sum_{v \in S} \sum_{u \in R \setminus S} W_{vu} - \sum_{v \in S} \sum_{u \in B \cap S} W_{vu} - \sum_{v \in S} \sum_{u \in B \setminus S} W_{vu}. \end{aligned} \tag{8}$$

We have

$$\sum_{v \in S} \sum_{u \in R \cap S} W_{vu} = \sum_{v \in R \cap S} \sum_{u \in R \cap S} W_{vu} + \sum_{v \in B \cap S} \sum_{u \in R \cap S} W_{vu} = \sum_{\{u,v\} \subset S \cap R} 2W_{uv} + \sum_{v \in B \cap S} \sum_{u \in R \cap S} W_{vu}.$$

Similarly, $\sum_{v \in S} \sum_{u \in B \cap S} W_{vu} = \sum_{\{u,v\} \subset S \cap B} 2W_{uv} + \sum_{v \in B \cap S} \sum_{u \in R \cap S} W_{vu}$.
Substituting back into Equation 8, we get

$$\mathrm{dif}(S) = \sum_{\{u,v\} \subset S \cap R} (2W_{uv}) - \sum_{\{u,v\} \subset S \cap B} (2W_{uv}) + \sum_{u \in S} \sum_{v \in R \setminus S} W_{uv} - \sum_{u \in S} \sum_{v \in R \setminus B} W_{uv}.$$

This is now a sum of independent variables, so we can apply Theorem 16. Firstly,

$$\mathbf{E}\left[\mathrm{dif}(S)\right] = p\big|S\big|(\big|R\big| - \big|B\big|) - p(|S \cap R| - |S \cap B|) \geq pm(n - 2n_0 - 1). \tag{9}$$

Moreover, each W_{uv} takes values in $[0,1]$ (a range of length 1) and $2W_{uv}$ takes values in $[0,2]$ (a range of length 2), so the sum of squares of these lengths are

$$\begin{aligned} F &= 4\binom{\big|S \cap R\big|}{2} + 4\binom{\big|S \cap B\big|}{2} + \big|S\big|\big|R \setminus S\big| + \big|S\big|\big|B \setminus S\big| \\ &= \big|S\big|(n - 2 + \big|S\big|) - 4\,|S \cap R|\,|S \cap B| \leq m(n - 2 + m). \end{aligned}$$

By Hoeffding's inequality:

$$\begin{aligned} \mathbf{P}\left(S \subseteq B_1 \mid R_0, B_0\right) &\leq \mathbf{P}\left(\mathrm{dif}(S) \leq 0\right) = \mathbf{P}\left(\mathrm{dif}(S) - \mathbf{E}\left[\mathrm{dif}(S)\right] \leq -\mathbf{E}\left[\mathrm{dif}(S)\right]\right) \\ &\leq \exp\left(-\frac{\mathbf{E}\left[\mathrm{dif}(S)\right]^2}{F}\right) \leq \exp\left(-\frac{2p^2(n-2n_0-1)^2 m}{n-2+m}\right). \end{aligned}$$

Applying a double union bound over choices of S and (R, B), noting that there are $\binom{n}{n_0}\binom{n}{m} \leq 4^n/n$ choices, we have

$$\mathbf{P}\left(\exists(R,B).\exists S.\ \left|B\right| = n_0, \left|S\right| = m, S \subset B_1\right) \ \leq\ \frac{4^n}{n}\exp\left(-\frac{2p^2(n-2n_0-1)^2 m}{n+m-2}\right).$$

Taking the complement event, we get the desired result. ◀

Lemma 18 turns out to be sufficient for the remaining blocks of our shrinking argument. The following lemmas are direct corollaries of Lemma 18.

▶ **Lemma 19.** *For $p \in (0,1)$, $\varepsilon_1 \in \left(0, \frac{1}{2}\right)$ and $n \in \mathbb{N}_{>1}$, with probability at least*

$$1 - \frac{1}{n}\exp\left(-\tfrac{1}{2}p^3(2\varepsilon_1 n - 1)^2 + 2n\log 2\right) = 1 - P_3(n,p,\varepsilon_1),$$

G is such that any choice of the Blue camp of at most $\left(\frac{1}{2} - \varepsilon_1\right)n = n_2^B$ members shrinks to size at most $\frac{1}{3}p(n-1) = n_3^B$ the next day.

▶ **Lemma 20.** *With $C_1 = (3\log 2)^{1/2}$ defined in Section 1.6, let $p \in (0,1)$, $n \in \mathbb{N}$ and $\varepsilon_1, \varepsilon_2 > 0$, then with probability at least*

$$1 - \exp\left(-n\left[(1-2\varepsilon_1)\varepsilon_2 - \varepsilon_1\right] - \log n\right) = 1 - P_2(n, \varepsilon_2, \varepsilon_1), \tag{10}$$

G is such that any choice of the Blue camp of at most $\frac{n-1}{2} - \left(\frac{C_1+\varepsilon_2}{2p}\right)\sqrt{n} = n_1^B$ members shrinks to size at most $\left(\frac{1}{2} - \varepsilon_1\right)n = n_2^B$ the next day.

A routine calculation shows that for appropriate choices of values for ε_1 and ε_2 such that $\varepsilon_1 < \varepsilon_2/(1+2\varepsilon_2)$, the bound in Equation (10) tends to 1 as $n \to +\infty$. Detailed proofs of Lemmas 19 and 20 are in Appendix A.2. These lemmas and Lemma 17, together with Corollary 15 form the complete "chain of shrinking" for the number of Blue vertices to reach 0 in four days, hence wrapping up the proof of Theorem 6.

4 Conclusion

The majority dynamics scheme on a network of n individuals is a process where each person is assigned an initial color, which changes daily to match the majority among their neighbors. The main results in this paper reveal a surprising facts. When the underlying network is a random $G(n,p)$ graph, for any given constants p and ε, there is a constant $c = c(p,\varepsilon)$ such that if one color has $\frac{n}{2} + c$ members in the initial state, then with probability at least $1-\varepsilon$, it covers the whole network in just four days, regardless of n.

Our main result, Theorem 6, yields an explicit lower-bound based on n, p and c for the probability that the side with the initial majority wins. It has two important implications. The first is the *Power of Few* phenomenon (Theorem 2), which shows that when $p = 1/2$ and $\varepsilon = .1$, c can be set to just 6, meaning six extra people is all it takes to win a large election with overwhelming odds. The second is an asymptotic dependency between the ε, n and c (Theorem 3), which shows that for any fixed p, there is a constant $K(p)$ such that choosing n and c both large enough so that $K(p)\max\{n^{-1}, c^{-2}\} < \varepsilon$ will ensure that the winning probability is at least $1-\varepsilon$.

The main idea behind Theorem 6 involves shrinking the set of Blue, the side with the initial minority, from $(n/2 - c)$ to 0 members in the course of four days. This chain of shrinking goes from $(n/2 - c)$ through $(n/2 - \Omega(\sqrt{n}))$, $(1/2 - \Omega(1))n$ and $(1/2 - \Omega(1))pn$, eventually reaching 0 after Day 4. There is a small probability that the shrinking fails to occur on each day, and their sum is the bound we obtained in the theorem's statement.

Although the results in this paper only applies for dense $G(n,p)$ graphs, we do cover sparse graphs in a separate in-progress paper [14], where we obtain the Power of Few phenomenon for $p = \Omega((\log n)/n)$, and discuss the end result (other than a win) for lower values of p. We nevertheless included one of the main proven results of the upcoming paper (Theorem 5), and used it to prove the main theorem in the paper [8] by Fountoulakis, Kang and Makai in Appendix A.3.

References

1 Dan Alistarh, James Aspnes, David Eisenstat, Rati Gelashvili, and Ronald L. Rivest. Time-space trade-offs in population protocols. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '17, page 2560–2579, USA, 2017. Society for Industrial and Applied Mathematics.

2 Dan Alistarh, Rati Gelashvili, and Milan Vojnović. Fast and exact majority in population protocols. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, PODC '15, page 47–56, New York, NY, USA, 2015. Association for Computing Machinery. doi:10.1145/2767386.2767429.

3 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, September 2016. doi:10.1016/j.spa.2016.02.015.

4 Béla Bollobás. *Models of Random Graphs*, chapter 2, pages 34–50. Cambridge University Press, Cambridge, 2001. doi:10.1017/CBO9780511814068.

5 Peter Clifford and Aidan Sudbury. A model for spatial conflict. *Biometrika*, 60:581–588, 1973. doi:10.1093/biomet/60.3.581.

6 Morris H. DeGroot. Reaching a consensus. *Journal of the American Statistical Association*, 69(345):118–121, 1974. URL: http://www.jstor.org/stable/2285509.

7 Carl-Gustav Esseen. A moment inequality with an application to the central limit theorem. *Scandinavian Actuarial Journal*, 1956(2):160–170, 1956. doi:10.1080/03461238.1956.10414946.

8 Nikolaos Fountoulakis, Mihyun Kang, and Tamás Makai. Resolution of a conjecture on majority dynamics: rapid stabilisation in dense random graphs, 2019. arXiv:1910.05820.

9 Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. doi:10.1080/01621459.1963.10500830.

10 Svante Janson, Tomasz Luczak, and Andrzej Rucinski. *Preliminaries*, chapter 1, pages 1–23. John Wiley & Sons, Ltd, 2011. doi:10.1002/9781118032718.ch1.

11 Elchanan Mossel, Joe Neeman, and Omer Tamuz. Majority dynamics and aggregation of information in social networks. *Autonomous Agents and Multi-Agent Systems*, 28(3):408–429, May 2014. doi:10.1007/s10458-013-9230-4.

12 Elchanan Mossel and Omer Tamuz. Opinion exchange dynamics. *Probab. Surveys*, 14:155–204, 2017. doi:10.1214/14-PS230.

13 I. G. Shevtsova. An improvement of convergence rate estimates in the lyapunov theorem. *Doklady Mathematics*, 82(3):862–864, December 2010. doi:10.1134/S1064562410060062.

14 Linh Tran and Van Vu. Power of few: The general case. Paper in preparation.

15 Roman Vershynin. *Concentration of Sums of Independent Random Variables*, pages 11–37. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, Cambridge, 2018. doi:10.1017/9781108231596.006.

A Appendix

In this appendix we provided detailed proofs for claims and lemmas not proven in the main text. We begin with some useful general probabilistic lemmas that are used throughout these proofs.

A.1 Proofs of Theorems 2 and 3

Proof of Theorem 2. Assume Theorem 6. Observe that if the conditions in (1) hold for some value of n, then they hold for all larger values of n. Let $n = 300$, $\varepsilon_1 = .15$ and $\varepsilon_2 = .3$ (along with $p = 1/2$ and $c = 6$), we have the condition in Equation (1) satisfied. Furthermore, a routine calculation shows that

$$P_1 \leq .08454, \quad P_2 < .00001, \quad P_3 < .00001, \quad P_4 \leq .00001,$$

which implies that $\mathbf{P}(B_4 \neq \varnothing) < 0.1$ or equivalently that Red wins in the fourth day with probability at least .9 (conditioned on the event $|B_0| = n^B \leq \frac{n}{2} - c$). ◀

Proof of Theorem 3. In this proof, only n and $c = c_n$ can vary. We can assume, without loss of generality, that $c_n \leq n/2$. Assuming Theorem 6, we choose (constants) $\varepsilon_1, \varepsilon_2$ such that $\varepsilon_2(1 - 2\varepsilon_1) - \varepsilon_1 > 0$, then a routine calculation shows that $P_2, P_3, P_4 = \mathrm{o}(n^{-2})$ and $P_1 = \Omega(n^{-1})$, so $P_1 + P_2 + P_3 + P_4 = P_1 + \mathrm{o}(1)$. We have, for sufficiently large c_n and sufficiently large n,

$$\sqrt{n-1}\,\Phi_0\left(\tfrac{2pc_n + \min\{p,1-p\}}{\sigma\sqrt{n-1}}\right) - \tfrac{C_0(1-2\sigma^2)}{\sigma} - \tfrac{C_1+\varepsilon_2}{2p} - \tfrac{1}{2\sqrt{n}} \geq \tfrac{\sqrt{n}}{2}\,\Phi_0\left(\tfrac{2c_n\sqrt{p}}{\sqrt{n}}\right) = \tfrac{T(n)}{2},$$
$$1/4 + 4C_0^2(1-2\sigma^2)^2\tfrac{n-1}{n-2} \leq 1/4 + 4 \cdot 0.6^2 \cdot 1.5 < 3.$$

Thus, $P_1 \leq 12T(n)^{-2}$. It then suffices to show $T(n) \geq H(p)\min\{c_n^{-2}, n^{-1}\}$ for some term $H(p)$ depending solely on p. Consider 2 cases:

If $c_n \geq \sqrt{n}$, then: $T(n) \geq \sqrt{n} \cdot \Phi_0(2\sqrt{p}) \geq \sqrt{n} \cdot 2\sqrt{p} \cdot \dfrac{\Phi_0(2)}{2} = \sqrt{p}\,\Phi_0(2)\sqrt{n}$.

If $c_n < \sqrt{n}$, then: $T(n) \geq \sqrt{n} \cdot \dfrac{2c_n\sqrt{p}}{\sqrt{n}} \cdot \dfrac{\Phi_0(2)}{2} = \sqrt{p}\,\Phi_0(2)\,c_n$.

In any case, $T(n) \geq H(p)\min\{c_n^{-2}, n^{-1}\}$, for $H(p) = \Phi_0(2)\sqrt{p}$, as desired. ◀

A.2 Proofs for lemmas in Day Two

We provide proofs for Lemmas 19 and 20 in Section 3.

Proof. In a $G(n,p)$ graph, $d(v)$ is a sum of $(n-1)$ $\mathrm{Bin}(1,p)$ random variables, so Theorem 16 implies that for any $u \in V$,

$$\mathbf{P}\left(d(u) \leq \tfrac{2}{3}p(n-1)\right) \leq \mathbf{P}\left(d(u) - \mathbf{E}\left[d(u)\right] \leq -\tfrac{1}{3}p(n-1)\right) \leq \exp\left(-\tfrac{2}{9}p^2(n-1)\right).$$

By a union bound, the probability that all vertices have more than $\frac{2}{3}p(n-1)$ neighbors is at least $1 - n\exp\left(-\frac{2}{9}p^2(n-1)\right) = 1 - P_4(n,p)$. Given this, a Blue camp of size $\frac{1}{3}p(n-1)$ surely vanishes the next day since it cannot form a majority in any vertex's neighborhood. The result then follows. ◀

Proof of Lemma 19. Let $n_2 \stackrel{\text{def}}{=} \left\lfloor\left(\frac{1}{2} - \varepsilon_1\right)n\right\rfloor$ and $m \stackrel{\text{def}}{=} \lceil\frac{1}{3}p(n-1)\rceil$. Lemma 18 implies that the G satisfies that every Blue set of at most n_2 vertices shrinks to size $m-1$ with probability at least

$$1 - \frac{4^n}{n}\exp\left(-\frac{2p^2(n-2n_2-1)^2m}{n+m-2}\right) = 1 - \frac{1}{n}\exp\left(-\frac{2p^2(n-2n_2-1)^2m}{n+m-2} + 2n\log 2\right).$$

Since $n_2 \le \left(\frac{1}{2} - \varepsilon_1\right) n$, $(n - 2n_2 - 1)^2 \ge (2\varepsilon_1 n - 1)^2$. Furthermore, $m \ge \frac{1}{3}p(n-1)$ so $\frac{m}{n+m-2} \ge \frac{p(n-1)/3}{n+p(n-1)/3-1} = \frac{p}{3+p} \ge \frac{p}{4}$. Therefore

$$\frac{1}{n}\exp\left(-\frac{2p^2(n-2n_2-1)^2 m}{n+m-2} + 2n\log 2\right) \le \frac{1}{n}\exp\left(-\frac{1}{2}p^3(2\varepsilon_1 n - 1)^2 + 2n\log 2\right).$$

The result then follows. ◀

Proof of Lemma 20. Let $n_2 \stackrel{\text{def}}{=} \left\lfloor \frac{n-1}{2} - \left(\frac{C_1+\varepsilon_2}{2p}\right)\sqrt{n} \right\rfloor$ and $m \stackrel{\text{def}}{=} \left\lceil \left(\frac{1}{2} - \varepsilon_1\right) n \right\rceil$. Lemma 18 implies that the G satisfies that every Blue set of at most n_2 vertices shrinks to size $m-1$ with probability at least

$$1 - \frac{4^n}{n}\exp\left(-\frac{2p^2(n-2n_2-1)^2 m}{n+m-2}\right) = 1 - \frac{1}{n}\exp\left[-\left(\frac{2p^2(n-2n_2-1)^2 m}{n+m-2} - 2n\log 2\right)\right].$$

Since $\frac{m}{n+m-2} \ge \frac{m}{n+m} \ge \frac{1-2\varepsilon_1}{3-2\varepsilon_1}$ and $n - 2n_2 - 1 \ge \left(\frac{C_1+\varepsilon_2}{p}\right)\sqrt{n}$, we can bound the exponent of the RHS of the above as follows

$$\begin{aligned}
&\frac{2p^2(n-2n_2-1)^2 m}{n+m-2} - 2n\log 2 \;\ge\; 2p^2\left(\frac{C_1+\varepsilon_2}{p}\right)^2 \frac{1-2\varepsilon_1}{3-2\varepsilon_1} n - \frac{2C_1^2}{3}n \\
&= \frac{2n}{3-2\varepsilon_1}\left[(C_1+\varepsilon_2)^2(1-2\varepsilon_1) - \frac{C_1^2}{3}(3-2\varepsilon_1)\right] \;\ge\; \frac{2n}{3}\left[\varepsilon_2(\varepsilon_2+2C_1)(1-2\varepsilon_1) - \frac{4}{3}\varepsilon_1 C_1^2\right] \\
&\ge \frac{2n}{3}\left[2.8\varepsilon_2(1-2\varepsilon_1) - 2.8\varepsilon_1\right] \;=\; \frac{5.6n}{3}\left[\varepsilon_2(1-2\varepsilon_1) - \varepsilon_1\right] \;\ge\; n\left[\varepsilon_2(1-2\varepsilon_1) - \varepsilon_1\right].
\end{aligned}$$

Note that we have used the facts that $C_1 > 1.4$ and $\log_2 < .7$. The proof is complete. ◀

A.3 Proof of Fountoulakis et al's Theorem from Theorem 5

We provide the proof of the main theorem in [8] (Theorem 4) with our Theorem 5.

Proof. Assume Theorem 5. Let R_0 and B_0 respectively be the initial Red and Blue camps. Fix a constant $0 < c' \le \varepsilon/6$. $\left|R_0\right| \sim \mathrm{Bin}(n, 1/2)$ since it is a sum of $\mathrm{Bin}(1, 1/2)$ variables. An application of the Berry-Esseen theorem (Theorem 10; with $C_0 = .56$) implies that

$$\mathbf{P}\left(\left|R_0\right| - \frac{n}{2} \le c'\sqrt{n}\right) \le \Phi(2c') + \frac{C_0}{\sqrt{n}} \;\text{ and }\; \mathbf{P}\left(\left|R_0\right| - \frac{n}{2} \le -c'\sqrt{n}\right) \ge \Phi(-2c') - \frac{C_0}{\sqrt{n}},$$

Thus

$$\begin{aligned}
\mathbf{P}\left(\left|\left|R_0\right| - \frac{n}{2}\right| \le c'\sqrt{n}\right) &\le \left(\Phi(2c') + \frac{C_0}{\sqrt{n}}\right) - \left(\Phi(-2c') - \frac{C_0}{\sqrt{n}}\right) \\
&\le \Phi(-2c', 2c') + \frac{2C_0}{\sqrt{n}} \;\le\; \frac{4c'}{\sqrt{2\pi}} + \frac{2C_0}{\sqrt{n}} \;\le\; \frac{\varepsilon}{3} + \frac{2C_0}{\sqrt{n}} \le \varepsilon/2,
\end{aligned}$$

for sufficiently large n.

On the other hand, if $\left|\left|R_0\right| - n/2\right| > c'\sqrt{n}$, then one of the sides has more than $n/2 + c'\sqrt{n}$ initial members, which we call the *majority side*. Now we apply Theorem 5 with ε replaced by $\varepsilon/2$. Notice that in the setting of Theorem 4 if we have $p = \lambda n^{-1/2}$ for λ sufficiently large, then $c'\sqrt{n} \ge c/p$, where c is the constant in Theorem 5. Thus, by this theorem, the probability for the majority side to win is at least $1 - \varepsilon/2$, and we are done by the union bound. ◀

Time-Space Tradeoffs for Distinguishing Distributions and Applications to Security of Goldreich's PRG

Sumegha Garg
Department of Computer Science, Princeton University, NJ, USA
sumegha.garg@gmail.com

Pravesh K. Kothari[1]
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
kotpravesh@gmail.com

Ran Raz
Department of Computer Science, Princeton University, NJ, USA
ran.raz.mail@gmail.com

Abstract

In this work, we establish lower-bounds against memory bounded algorithms for distinguishing between natural pairs of related distributions from samples that arrive in a streaming setting.

Our first result applies to the problem of distinguishing the uniform distribution on $\{0,1\}^n$ from uniform distribution on some unknown linear subspace of $\{0,1\}^n$. As a specific corollary, we show that any algorithm that distinguishes between uniform distribution on $\{0,1\}^n$ and uniform distribution on an $n/2$-dimensional linear subspace of $\{0,1\}^n$ with non-negligible advantage needs $2^{\Omega(n)}$ samples or $\Omega(n^2)$ memory (tight up to constants in the exponent).

Our second result applies to distinguishing outputs of Goldreich's *local* pseudorandom generator from the uniform distribution on the output domain. Specifically, Goldreich's pseudorandom generator $\mathcal{G}$ fixes a predicate $P : \{0,1\}^k \to \{0,1\}$ and a collection of subsets $S_1, S_2, \ldots, S_m \subseteq [n]$ of size k. For any seed $x \in \{0,1\}^n$, it outputs $P(x_{S_1}), P(x_{S_2}), \ldots, P(x_{S_m})$ where x_{S_i} is the projection of x to the coordinates in S_i. We prove that whenever P is t-resilient (all non-zero Fourier coefficients of $(-1)^P$ are of degree t or higher), then no algorithm, with $< n^\epsilon$ memory, can distinguish the output of $\mathcal{G}$ from the uniform distribution on $\{0,1\}^m$ with a large inverse polynomial advantage, for stretch $m \leq \left(\frac{n}{t}\right)^{\frac{(1-\epsilon)}{36}\cdot t}$ (barring some restrictions on k). The lower bound holds in the streaming model where at each time step i, $S_i \subseteq [n]$ is a randomly chosen (ordered) subset of size k and the distinguisher sees either $P(x_{S_i})$ or a uniformly random bit along with S_i.

An important implication of our second result is the security of Goldreich's generator with super linear stretch (*in the streaming model*), against memory-bounded adversaries, whenever the predicate P satisfies the *necessary* condition of t-resiliency identified in various prior works.

Our proof builds on the recently developed machinery for proving time-space trade-offs (Raz 2016 and follow-ups). Our key technical contribution is to adapt this machinery to work for *distinguishing* problems in contrast to prior works on similar results for *search*/learning problems.

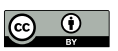
2012 ACM Subject Classification Theory of computation → Pseudorandomness and derandomization

Keywords and phrases memory-sample tradeoffs, bounded storage cryptography, Goldreich's local PRG, distinguishing problems, refuting CSPs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.21

Category RANDOM

[1] Part of this work was completed when Pravesh was at Princeton University and Institute for Advanced Study, Princeton.



© Sumegha Garg, Pravesh K. Kothari, and Ran Raz;
licensed under Creative Commons License CC-BY



Related Version A full version of this paper is available at `https://arxiv.org/abs/2002.07235`.

Funding *Ran Raz*: Research supported by the Simons Collaboration on Algorithms and Geometry, by a Simons Investigator Award and by the National Science Foundation grant No. CCF-1714779.

Acknowledgements We would like to thank Avishay Tal and David Woodruff for the discussions about the problem of distinguishing subspaces.

1 Introduction

This work is motivated by the following basic question: suppose an algorithm is provided with a stream of m i.i.d. samples from a random source. What's the minimum memory required to decide whether the source is "truly random" or "pseudorandom"?

Algorithmically distinguishing perfect randomness from pseudorandomness naturally arises in the context of learning theory (and can even be equivalent to learning in certain models [20, 24, 60, 36]), pseudorandomness and cryptography.

There has been a surge of progress in proving lower bounds for memory-bounded streaming algorithms beginning with Shamir [55] and Steinhardt-Valiant-Wager [57] who conjectured a $\Omega(n^2)$ memory lower bound for learning parity functions with $2^{o(n)}$ samples. This conjecture was proven in [53]. In a follow up work, this was generalized to learning sparse parities in [35] and more general learning problems in [54, 29, 45, 14, 18, 47, 46, 56, 30].

All of these lower bounds hold for learning (more generally, *search*) problems that ask to identify an unknown member of a target function class from samples. In this work, we build on the progress above and develop techniques to show lower bounds for apparently easier task of simply distinguishing uniformly distributed samples from pseudorandom ones. [25] studies the related problem of distribution testing under communication and memory constraints. [25] gave a one-pass streaming algorithm (and a matching lower bound for a broad range of parameters) for uniformity testing on $[N]$ that uses m memory and $O(N \log(N)/(m\epsilon^4))$ samples for distinguishing between uniform distribution on $[N]$ and any distribution that is ϵ-far from uniform.

As we next discuss, our results have consequences of interest in cryptography (ruling out memory-bounded attacks on Goldreich's pseudorandom generator [31] in the streaming model) and average-case complexity (unconditional lower bounds on the number of samples needed, for memory-bounded algorithms, to refute random constraint satisfaction problems, in the streaming model).

1.1 Our Results

We now describe our results in more detail. Our main results show memory-sample trade-offs for distinguishing between truly random and pseudorandom sources for the following two settings:

1. **Uniform vs k-Subspace Source**: The pseudorandom subspace source of dimension k chooses some arbitrary k-dimensional linear subspace $S \subseteq \{0,1\}^n$ and draws points uniformly from S. The truly random source draws points uniformly from $\{0,1\}^n$.
2. **Uniform vs Local Pseudorandom Source**: The pseudorandom source fixes a k-ary Boolean predicate $P : \{0,1\}^k \to \{0,1\}$. It chooses a uniformly random $x \in \{0,1\}^n$ and generates samples $(\alpha, b) \in [n]^{(k)} \times \{0,1\}$ where $[n]^{(k)}$ represents the set of all ordered k-tuples with exactly k elements from $[n]$ and α is chosen uniformly at random from $[n]^{(k)}$ and b is the evaluation of P at x^α - the k-bit string obtained by projecting x onto the coordinates indicated by α. The truly random source generates samples (α, b) where $\alpha \in [n]^{(k)}$ and $b \in \{0,1\}$ are chosen uniformly and independently.

We model our algorithm by a *read-once branching program* (ROBP) of width 2^b (or memory b) and length m. Such a model captures any algorithm that takes as input a stream of m samples and has a memory of at most b bits. Observe that there's no restriction on the computation done at any node of an ROBP.

Roughly speaking, this model gives the algorithm unbounded computational power and bounds only its memory-size and the number of samples used.

Our first main result shows a lower bound on memory-bounded ROBPs for distinguishing between uniform and k-subspace sources.

▶ **Theorem 1** (Uniform vs Subspace Sources). *Any algorithm that distinguishes between uniform and subspace source of dimension k (assuming $k > c \log n$ for some large enough constant c) with probability at least $1/2 + 2^{-o(k)}$ requires either a memory of $\Omega(k^2)$ or at least $2^{\Omega(k)}$ samples. In particular, distinguishing between the uniform distribution on $\{0,1\}^n$ and the uniform distribution on an unkown linear subspace of dimension $n/2$ in $\{0,1\}^n$ requires $\Omega(n^2)$ memory or $2^{\Omega(n)}$ samples.*

Crouch et. al. [16] recently proved that any algorithm that uses at most $n/16$ bits of space requires $\Omega(2^{n/16})$ samples to distinguish between uniform source and a subspace source of dimension $k = n/2$. They suggest the question of improving the space bound to $\Omega(n^2)$ while noting that their techniques do not suffice. For $k = \Theta(n)$, our lower bound shows that any algorithm with memory at most cn^2 for some absolute constant c requires $2^{\Omega(n)}$ samples. This resolves their question.

Upper bound. In Section 4, we exhibit a simple explicit branching program that uses $2^{O(k)}$ samples and $O(1)$ memory to succeed in solving the distinguishing problem with probability $3/4$. We also show a simple algorithm that uses $O(k^2)$ memory and $O(k)$ samples, and succeeds in solving the distinguishing problem with probability $3/4$. *Thus, in the branching program model, the lower bound is tight up to constants in the exponent.*

Our second main result gives a memory-sample trade-off for the uniform vs local pseudorandom source problem for all predicates that have a certain well-studied pseudorandom property studied in cryptography under the name of *resilience*.

A k-ary Boolean function P is said to be *t-resilient* if t is the maximum integer such that $(-1)^P$ (taking the range of the boolean function to be {-1,1}) has zero correlation with every parity function of at most $t-1$ out of k bits. In particular, the parity function on k bits is k-resilient.

▶ **Theorem 2** (Uniform vs Local Pseudorandom Sources). *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be a t-resilient k-ary predicate for $k < n^{(1-\epsilon)/6}/3, n/c^2$. Then, any ROBP that succeeds with probability at least $1/2 + \Omega(\left(\frac{t}{n}\right)^{\Omega(t\cdot(1-\epsilon))})$ at distinguishing between uniform and local pseudorandom source for predicate P, requires $\left(\frac{n}{t}\right)^{\Omega(t\cdot(1-\epsilon))}$ samples or n^ϵ memory.*

Upper bound. In Subsection 5.3, we give an algorithm that takes $(n^\epsilon + k)k \log n$ memory and $(n^{(1-\epsilon)k})(n^\epsilon + k)$ samples, and distinguishes between uniform and local pseudorandom source for any predicate P, with probability $99/100$. Thus, the lower bounds are almost tight up to $\log n$ factors and constant factors in the exponent for certain predicates ($t = \Omega(k)$). The question of whether there exists a better algorithm that runs in $O(n^{(1-\epsilon)t})$ samples and $O(n^\epsilon)$ memory, and distinguishes between uniform and local pseudorandom source with high probability, for t-resilient predicates P, remains open.

[2] c is a large enough constant

This result has interesting implications for well-studied algorithmic questions in average-case complexity and cryptography such as refuting random constraint satisfaction [27, 2, 52, 37]) and existence of local pseudorandom generators [17, 49, 11, 43, 4, 5] with super linear stretch where a significant effort has focused on proving lower bounds on various restricted models such as propositional and algebraic proof systems, spectral methods, algebraic methods and semidefinite programming hierarchies. While bounded memory attacks are well-explored in cryptography [44, 15, 10, 9, 59, 26, 53, 61, 32, 58, 34, 19], to the best of our knowledge, memory has not been studied as explicit resource in this context. We discuss these applications further in the paper.

For the special case when $P(x) = \sum_i^k x^i \mod 2$, the parity function on k bits, we can prove stronger results for a wider range of parameters.

▶ **Theorem 3** (Uniform vs Local Pseudorandom Sources with Parity Predicate). *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be the parity predicate on k bits for $0 < k < n/c$ (c is a large enough constant). Suppose there's a ROBP that distinguishes between uniform and local pseudorandom source for the parity predicate, with probability at least $1/2 + s$ and uses $< n^\epsilon$ memory. If $s > \Omega\left(\left(\frac{k}{n}\right)^{\Omega((1-\epsilon)\cdot k)}\right)$, then, the ROBP requires $\left(\frac{n}{k}\right)^{(\Omega((1-\epsilon)\cdot k)}$ samples.*

A recent concurrent work [19] builds symmetric encryption schemes which are secure against memory bounded adversaries, where each ciphertext makes at most k calls to a "random" function. Even though the applications and techniques are different, the setting of the analysis is almost identical and the paper obtains better bounds (in terms of constants in the exponent) for the (and only the) parity predicate on k bits.

The above results show lower bounds for *sublinear* memory algorithms. For a slight variant of the above uniform vs local pseudorandom source problem, we can in fact upgrade our results to obtain the following lower bounds against *super-linear* memory algorithms. See Section 4 for details.

▶ **Theorem 4.** *For large enough n and $k > c \log n$ (where c is a large enough constant) and $k \leq \frac{n}{4}$, any algorithm that can distinguish satisfiable sparse parities of sparsity k on n variables (of type $(a, b) = (a^1, a^2, ..., a^n, b) \in \{0,1\}^{n+1}$, where $\forall i \in [n]$, $a^i = 1$ with probability $\frac{k}{n}$ and $b = \langle a, x \rangle$) from random ones (of similar type (a, b) but b is now chosen uniformly at random from $\{0,1\}$), with success probability at least $\frac{1}{2} + 2^{-o(k)}$, requires either a memory of size $\Omega(nk)$ or $2^{\Omega(k)}$ samples.*

In Remark 15, we observe that the above theorem is almost tight. Specifically, we observe that there are ROBPs that use a constant memory and $O(n2^{O(k)})$ samples or $O(nklogn)$ memory and $O(n)$ samples to distinguish uniform sources from locally pseudorandom ones.

1.2 Applications to Security of Goldreich's Pseudorandom Generator

A fundamental goal in cryptography is to produce secure constructions of cryptographic primities that are highly efficient. In line with this goal, Goldreich [31] proposed a candidate one-way function given by the following pseudorandom mapping that takes n-bit input x and outputs m bits: fix a *predicate* $P : \{0,1\}^k \to \{0,1\}$, pick $a_1, a_2, \ldots, a_m$ uniformly at random[3] from $[n]^{(k)}$ and output $P(x^{a_1}), P(x^{a_2}), \ldots, P(x^{a_m})$. Here, $a_1, \ldots, a_m$ and P are public and the seed x is secret. Later works (starting with [48]) suggested using this candidate as pseudorandom generator.

[3] More generally, Goldreich proposed that $a_1, a_2, \ldots, a_m$ could be chosen in a pseudorandom way so as to ensure a certain "expansion" property. We omit a detailed discussion here.

The main question of interest is the precise trade-off between the *locality* k and the *stretch* m for a suitable choice of the predicate P. In several applications, we need that the generator has a super-linear stretch (i.e. $m = n^{1+\delta}$ for some $\delta > 0$) with constant locality (i.e. $k = O(1)$).

The simplicity and efficiency of such a candidate is of obvious appeal. This simplicity has been exploited to yield a host of applications including public-key cryptography from combinatorial assumptions [6], highly efficient secure multiparty computation [33] and most recently, basing *indistinguishability obfuscation* on milder assumptions [38, 3, 41, 39, 40].

Evidence for the security of Goldreich's candidate has been based on analyzing natural classes of attacks based on propositional proof systems [1], spectral methods and semidefinite programming hierarchies [51, 2, 12, 37, 42, 11] and algebraic methods [7, 8]. In particular, previous works [37, 8] identified t-resiliency of the predicate P as a necessary condition for the security of the candidate for $m = n^{\Omega(t)}$ stretch.

The *uniform vs local pseudorandom source* problem considered in this work is easily seen as the algorithmic question of distinguishing the output stream generated by Goldreich's candidate generator from a uniformly random sequence of bits. In particular, our results imply security of Goldreich's candidate against bounded memory algorithms for super-linear stretch when instantiated with any t-resilient predicate for large enough constant t (but in the streaming model). Goldreich's candidate generator would fix the sets $a_1, a_2, \ldots, a_m$ (which are public) and output $P(x^{a_1}), P(x^{a_2}), \ldots, P(x^{a_m})$ for n sized input x ($m > n$). We prove the security of Goldreich's generator in the model where $a_1, a_2, \ldots, a_m$, still public, are chosen uniformly at random from $[n]^{(k)}$ and streamed with the generated bits.

We note that our lower bounds continue to hold even when the locality k grows polynomially with the seed length n.

▶ **Corollary 5** (Corollary of Theorem 2). *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be a t-resilient k-ary predicate for $k = O(n^{(1-\epsilon)/6})$. Then, Goldreich's PRG, when instantiated with any t-resilient k-ary predicate P such that $k \geq t > 36$ and stretch $m = (n/t)^{O(t)(1-\epsilon)}$, is secure against all read-once branching programs with memory-size bounded from above by n^{ϵ}, in the streaming model.*

1.3 Applications to Refuting Random CSPs

Theorems 2 and 3 can also be intepreted as lower bounds for the problem of refuting random constraint satisfaction problems.

A random CSP with predicate $P : \{0,1\}^k \to \{0,1\}$ is a constraint satisfaction problem on n variables $x \in \{0,1\}^n$. More relevant to us is the variant where the constraints are randomly generated as follows: choose an ordered k-tuple of variables a from $[n]$ at random, a bit $b \in \{0,1\}$ at random and impose a constraint $P(x^a) = b$. When the number of constraints $m \gg n$, the resulting instance is unsatisfiable with high probability for any non-constant predicate P. The natural algorithmic problem in this regime is that of *refutation* - efficiently finding a short witness that certifies that the given instance is far from satisfiable. It is then easy to note that the uniform vs local pseudorandom source problem is the task of distinguishing between constraints in a random CSP (with predicate P) and one with a satisfying assignment. Note that refutation is formally harder than the task of distinguishing between a random CSP and one that has a satisfying assignment.

Starting with investigating in proof complexity, random CSPs have been intensively studied in the past three decades. When P is t-resilient for $t \geq 3$, all known efficient algorithms [2] require $m \gg n^{1.5}$ samples for the refutation problem. This issue was brought

to the forefront in [27] where Feige made the famous "Feige's Hypothesis" conjecturing the impossibility of refuting random 3SAT in polynomial time with $\Theta(n)$ samples. Variants of Feige's hypothesis for other predicates have been used to derive hardness results in both supervised [21, 22, 23] and unsupervised machine learning [13].

In [51], t-resilience was noted as a necessary condition for the refutation problem to be hard. Our Theorems 2 and 3 confirm this as a sufficient condition for showing lower-bounds for the refutation (in fact, even for the easier "distinguishing" variant) of random CSPs, with t-resilient predicates, in the streaming model with bounded memory.

2 Preliminaries

Denote by log the logarithm to base 2. We use $Ber(p)$ to denote the Bernoulli distribution with parameter p (probability of being 1). We use $[n]$ to denote the set $\{1, 2, ..., n\}$.

For a random variable Z and an event E, we denote by $\mathbb{P}_Z$ the distribution of the random variables Z, and we denote by $\mathbb{P}_{Z|E}$ the distribution of the random variable Z conditioned on the event E.

Given an $n-$bit vector $y \in \{0,1\}^n$, we use y^i to denote the i^{th} coordinate of y, that is, $y = (y^1, y^2, ..., y^n)$. We use $y^{-i} \in \{0,1\}^{n-1}$ to denote y but with the ith coordinate deleted. Given two $n-$bit vectors y, y', we use $\langle y, y' \rangle$ to denote the inner product of y and y' modulo 2, that is, $\langle y, y' \rangle = \sum_{i=1}^{n} y^i y'^i \mod 2$. We use $|y|$ to denote the number of ones in the vector y.

Given a set S, we use $y \in_R S$ to denote the random process of picking y uniformly at random from the set S. Given a probability distribution D, we use $y \sim D$ to denote the random process of sampling y according to the distribution D.

Next, we restate (for convenience) the definitions and results from previous papers [53, 54, 35, 29] that we use.

Viewing a Learning Problem as a Matrix

Let X, A be two finite sets of size larger than 1.

Let $M : A \times X \to \{-1, 1\}$ be a matrix. The matrix M corresponds to the following learning problem: There is an unknown element $x \in X$ that was chosen uniformly at random. A learner tries to learn x from samples (a, b), where $a \in A$ is chosen uniformly at random and $b = M(a, x)$. That is, the learning algorithm is given a stream of samples, $(a_1, b_1), (a_2, b_2) \ldots$, where each a_t is uniformly distributed and for every t, $b_t = M(a_t, x)$.

These papers model the learner for the learning problem corresponding to the matrix M using a branching program:

▶ **Definition 6** (Branching Program for a Learning Problem). *A branching program of length m and width d, for learning, is a directed (multi) graph with vertices arranged in $m+1$ layers containing at most d vertices each. In the first layer, that we think of as layer 0, there is only one vertex, called the start vertex. A vertex of outdegree 0 is called a leaf. All vertices in the last layer are leaves (but there may be additional leaves). Every non-leaf vertex in the program has $2|A|$ outgoing edges, labeled by elements $(a, b) \in A \times \{-1, 1\}$, with exactly one edge labeled by each such (a, b), and all these edges going into vertices in the next layer. Each leaf v in the program is labeled by an element $\tilde{x}(v) \in X$, that we think of as the output of the program on that leaf.*

Computation-Path: *The samples $(a_1, b_1), \ldots, (a_m, b_m) \in A \times \{-1, 1\}$ that are given as input, define a computation-path in the branching program, by starting from the start vertex and following at step t the edge labeled by (a_t, b_t), until reaching a leaf. The program outputs the label $\tilde{x}(v)$ of the leaf v reached by the computation-path.*

Success Probability: *The success probability of the program is the probability that $\tilde{x} = x$, where $\tilde{x}$ is the element that the program outputs, and the probability is over $x, a_1, \ldots, a_m$ (where x is uniformly distributed over X and $a_1, \ldots, a_m$ are uniformly distributed over A, and for every t, $b_t = M(a_t, x)$).*

▶ **Theorem 7** ([53, 54, 29]). *Any branching program that learns $x \in \{0,1\}^n$, from random linear equations over $\mathbb{F}_2$ with success probability 2^{-cn} requires either a width of $2^{\Omega(n^2)}$ or a length of $2^{\Omega(n)}$ (where c is a small enough constant).*

▶ **Theorem 8** ([29]). *Any branching program that learns $x \in \{0,1\}^n$, from random sparse linear equations, of sparsity exactly ℓ, over $\mathbb{F}_2$ with success probability 2^{-cl} (where c is a small enough constant) requires:*

1. *Assuming $\ell \leq n/2$: either a width of size $2^{\Omega(n \cdot \ell)}$ or length of $2^{\Omega(\ell)}$.*
2. *Assuming $\ell \leq n^{0.9}$: either a width of size $\Omega(n \cdot \ell^{0.99})$ or length of $\ell^{\Omega(\ell)}$.*

Norms and Inner Products

Let $p \geq 1$. For a function $f : X \to \mathbb{R}$, denote by $\|f\|_p$ the ℓ_p norm of f, with respect to the uniform distribution over X, that is:

$$\|f\|_p = \left(\mathop{\mathbf{E}}_{x \in_R X} [|f(x)|^p] \right)^{1/p}.$$

For two functions $f, g : X \to \mathbb{R}$, define their inner product with respect to the uniform distribution over X as

$$\langle f, g \rangle = \mathop{\mathbf{E}}_{x \in_R X} [f(x) \cdot g(x)].$$

For a matrix $M : A \times X \to \mathbb{R}$ and a row $a \in A$, we denote by $M_a : X \to \mathbb{R}$ the function corresponding to the a-th row of M. Note that for a function $f : X \to \mathbb{R}$, we have $\langle M_a, f \rangle = \frac{(M \cdot f)_a}{|X|}$.

▶ **Definition 9** (L_2-Extractor, [29]). *Let X, A be two finite sets. A matrix $M : A \times X \to \{-1, 1\}$ is a (k', ℓ')-L_2-Extractor with error $2^{-r'}$, if for every non-negative $f : X \to \mathbb{R}$ with $\frac{\|f\|_2}{\|f\|_1} \leq 2^{\ell'}$ there are at most $2^{-k'} \cdot |A|$ rows a in A with*

$$\frac{|\langle M_a, f \rangle|}{\|f\|_1} \geq 2^{-r'}.$$

▶ **Lemma 10** ([35]). *Let T_l be the set of n-bit vectors with sparsity exactly-l for $l \in \mathbb{N}$, that is, $T_l = \{x \in \{0,1\}^n \mid \sum_{i=1}^n x^i = l\}$. Let $\delta \in (0, 1]$. Let $\mathcal{B}_{T_l}(\delta) = \{\alpha \in \{0,1\}^n \mid \left| \boldsymbol{E}_{x \in T_l} (-1)^{\langle \alpha, x \rangle} \right| > \delta\}$. Then, for $\delta \geq (\frac{8l}{n})^{\frac{l}{2}}$,*

$$|\mathcal{B}_{T_l}(\delta)| \leq 2e^{-\delta^{2/l} \cdot n/8} \cdot 2^n$$

Branching Program for a Distinguishing Problem

Let X, A be two finite sets of size larger than 1. Let D_0 be a distribution over the sample space $|A|$. Let $\{D_1(x)\}_{x \in X}$ be a set of distributions over the sample space $|A|$. Consider the following distinguishing problem: An unknown $\mathbf{b} \in \{0,1\}$ is chosen uniformly at random. If $\mathbf{b} = 0$, the distinguisher is given independent samples from D_0. If $\mathbf{b} = 1$, an unknown $x \in X$ is chosen uniformly at random, and the distinguisher is given independent samples from $D_1(x)$. The distinguisher tries to learn $\mathbf{b}$ from the samples drawn according to the respective distributions.

Formally, we model the distinguisher by a branching program as follows.

▶ **Definition 11** (Branching Program for a Distinguishing Problem). *A branching program of length m and width d, for distinguishing, is a directed (multi) graph with vertices arranged in $m+1$ layers containing at most d vertices each. In the first layer, that we think of as layer 0, there is only one vertex, called the start vertex. A vertex of outdegree 0 is called a leaf. All vertices in the last layer are leaves (but there may be additional leaves). Every non-leaf vertex in the program has $|A|$ outgoing edges, labeled by elements $a \in A$, with exactly one edge labeled by each such a, and all these edges going into vertices in the next layer. Each leaf v in the program is labeled by a $\tilde{\boldsymbol{b}}(v) \in \{0,1\}$, that we think of as the output of the program on that leaf.*

Computation-Path: *The samples $a_1, \ldots, a_m \in A$ that are given as input, define a computation-path in the branching program, by starting from the start vertex and following at step t the edge labeled by a_t, until reaching a leaf. The program outputs the label $\tilde{\boldsymbol{b}}(v)$ of the leaf v reached by the computation-path.*

Success Probability: *The success probability of the program is the probability that $\tilde{\boldsymbol{b}} = \boldsymbol{b}$, where $\tilde{\boldsymbol{b}}$ is the element that the program outputs, and the probability is over $\boldsymbol{b}, x, a_1, \ldots, a_m$ (where $\boldsymbol{b}$ is uniformly distributed over $\{0,1\}$, x is uniformly distributed over X and $a_1, \ldots, a_m$ are independently drawn from D_0 if $\boldsymbol{b} = 0$ and $D_1(x)$ if $\boldsymbol{b} = 1$).*

3 Overview of the Proofs

We prove our theorems using two different techniques. We prove Theorems 1 and 4 through reductions to the memory-sample lower bounds for the corresponding learning problems in Section 4. Informally, for Theorem 1, we construct a branching program that learns the unknown vector x from random linear equations in $\mathbb{F}_2$ by guessing each bit one by one sequentially and using the distinguisher, for distinguishing subspaces from uniform, to check if it guessed correctly. Then, we are able to lift the previously-known memory-sample lower bounds for the learning problem (Theorem 7) to the distinguishing problem. Similarly, we lift the memory-sample lower bounds for a variant of the learning problem in Theorem 8 to the get Theorem 4.

Theorems 2 and 3 are restated in Section 5. A brief overview of these theorems are as follows. Recall, a pseudorandom source fixes a k-ary Boolean predicate $P : \{0,1\}^k \to \{0,1\}$. It chooses a uniformly random $x \in \{0,1\}^n$ and generates samples $(\alpha, b) \in [n]^{(k)} \times \{0,1\}$ where α is a uniformly random (ordered) k-tuple of indices in $[n]$ and b is the evaluation of P at x^α - the k-bit string obtained by projecting x onto the coordinates indicated by α. The truly random source samples (α, b) where $\alpha \in [n]^{(k)}$ and $b \in \{0,1\}$ are chosen uniformly and independently. The problem for a distinguisher is to correctly guess whether the m samples are generated by a pseudorandom or a uniform source, when the samples arrive in a stream. We first show through a *hybrid argument* that a distinguisher A that distinguishes between the uniform and pseudorandom source, with an advantage of s over $1/2$, can also distinguish (with advantage of at least s/m) when only the jth (for some j) sample is drawn from the "unknown source", the first $j-1$ samples are drawn from the pseudorandom source and the last $m-j$ samples are drawn from the uniform source.

Let v be the memory state of A after seeing the first $j-1$ samples, which were generated from a pseudorandom source with a seed x picked uniformly at random from $\{0,1\}^n$. Let $\mathbb{P}_{x|v}$ be the probability distribution of the random variable x conditioned on reaching v. If the jth sample is generated using the same pseudorandom source, then $\forall \alpha \in [n]^{(k)}$, the bit b is 0 with probability $\sum_{x' : P(x'^\alpha)=0} \mathbb{P}_{x|v}(x')$ and 1 with probability $1 - \sum_{x' : P(x'^\alpha)=0} \mathbb{P}_{x|v}(x')$. If the jth sample is generated using the uniform source, then $\forall \alpha \in [n]^{(k)}$, the bit b is 0 with

probability 1/2 and 1 with probability 1/2. Thus, for any α, A can identify the "unknown source" with an at most $\left|\sum_{x':P(x'^{\alpha})=0} \mathbb{P}_{x|v}(x') - 1/2\right|$ advantage. We show that when A has low memory ($< n^{\epsilon}$ for some $0 < \epsilon < 1$), then with high probability, it reaches a state v such that $\mathbb{P}_{x|v}$ has high min-entropy (informally, it's hard to determine the seed for the pseudorandom source). The argument till now (last 2 paragraphs) is common and has been previously used to prove that a "source" fools bounded space, such as in [50].

Next, we use t-resiliency of P to show that when $\mathbb{P}_{x|v}$ has high min-entropy, then with high probability over $\alpha \in [n]^{(k)}$, b behaves almost like in a uniform source (Lemma 19), that is, $|\sum_{x':P(x'^{\alpha})=0} \mathbb{P}_{x|v}(x') - 1/2|$ is small. Hence, with high probability, it's hard for A to judge with 'good' advantage whether b was generated from a pseudorandom or a uniform source. Note that the last $m - j$ samples generated by a uniform source can't better this advantage.

4 Time-Space Tradeoff through Reduction to Learning

In this section, we will state time-space tradeoffs for the following distinguishing problems, which are proved using black-box reduction from the corresponding learning problems.

Distinguishing Subspaces from Uniform

Informally, we study the problem of distinguishing between the cases when the samples are drawn from a uniform distribution over $\{0,1\}^n$ and when the samples are drawn randomly from a subspace of rank k over $\mathbb{F}_2$. Let $L(k,n)$ be the set of all linear subspaces of dimension k ($\subseteq \{0,1\}^n$), that is, $L(k,n)$ contains all subspaces V such that $V = \{v \in \{0,1\}^n \mid \langle w_i, v\rangle = 0 \ \forall i \in [n-k]\}$ for some linearly independent vectors $w_1, w_2, ..., w_{n-k}$. Formally, we consider distinguishers for distinguishing between the following distributions:

1. D_0: Uniform distribution over $\{0,1\}^n$.
2. $D_1(S)$, $S \in L(k,n)$: Uniform distribution over S.

Note: If the subspace S is revealed, it's easy for a branching program of constant width to distinguish w.h.p. by checking the inner product of the samples with a vector in the orthogonal complement of S.

A distinguisher can distinguish subspaces if for an unknown random linear subspace $S \in L(k,n)$, it can distinguish between D_0 and $D_1(S)$. Formally, a distinguisher L, after seeing m samples, has a success probability of p if

$$\frac{\Pr_{u_1,...,u_m \sim D_0}[L(u_1,...,u_m)=0] + \Pr_{S \in_R L(k,n); u_1,...,u_m \sim D_1(S)}[L(u_1,...,u_m)=1]}{2} = p \quad (1)$$

▶ **Theorem 12.** *For $k > c_2 \log n$ (where c_2 is a large enough constant), any algorithm that can distinguish k-dimensional subspaces over $\mathbb{F}_2^n$ from $\mathbb{F}_2^n$ ($\{0,1\}^n$), when samples are drawn uniformly at random from the subspace or $\mathbb{F}_2^n$ respectively, with success probability at least $\frac{1}{2} + 2^{-o(k)}$ requires either a memory of size $\Omega(k^2)$ or $2^{\Omega(k)}$ samples.*

For the proof, refer to the full version of the paper [28]. Briefly, we prove that using a distinguisher for distinguishing subspaces, we can construct a branching program that learns an unknown bit vector x from random linear equations over $\mathbb{F}_2$. Then, we are able to lift the time-space tradeoffs of Theorem 7.

▶ Remark 13 (Tightness of the Lower Bound). We note two easy upper bounds that show that our results in Theorem 12 are tight (up to constants in the exponent). Firstly, we observe an algorithm B_1 that distinguishes subspaces of dimension k from uniform, using $O(k^2)$ memory and $O(k)$ samples, with probability at least 3/4 ($0 < k \leq n-1$). B_1 stores

the first $\min(8k, n)$ bits of the first $8k$ samples (in $O(k^2)$ memory); outputs 1 if the samples (projected onto the first $\min(8k, n)$ coordinates) belong to a $\leq k$-dimensional subspace (of $\{0,1\}^{\min(8k,n)}$), and 0 otherwise (can be checked using gaussian elimination). When the samples are drawn from $D_1(S)$ for some k-dimensional subspace S, then B_1 always outputs the correct answer. When the samples are drawn from a uniform distribution on $\{0,1\}^n$, the probability that $8k$ samples form a k-dimensional subspace is at most

$$\binom{8k}{k} \cdot \frac{1}{2^{7k}} \leq (8e)^k 2^{-7k} < 2^{-2k} \leq 1/4$$

(because, if the $8k$ samples form a k-dimensional subspace, then at least $7k$ of them are linearly dependent on the previously stored samples and that happens with at most $1/2$ probability for each sample). Hence, B_1 errs with at most $1/4$ probability.

Secondly, we observe that there exists a branching program that distinguishes subspaces of dimension k from uniform using constant width and $O(k \cdot 2^k)$ length with probability at least $3/4$. Before, we show a randomized algorithm P that distinguishes between D_0 and $D_1(S)$ for every $S \in L(k, n)$ with high probability. P is described as follows:

1. Repeat steps 2 to 3 sequentially for $t = 10 \cdot 2^k$ iterations.
2. Pick a non-zero vector v uniformly at random from $\{0,1\}^n$. For the next $2k$ samples (of the form $a \in \{0,1\}^n$), check if $\langle a, v \rangle = 0$.
3. If all the $2k$ samples are orthogonal to v, exit the loop and output 1.
4. Output 0 (None of the randomly chosen vectors were orthogonal to all the samples seen in its corresponding iteration).

The number of samples seen by P is $20k \cdot 2^k$. Now, we prove that for every subspace S of dimension k, that is, $S \in L(k, n)$, P distinguishes between D_0 and $D_1(S)$ with probability at least $1 - \frac{1}{2}(e^{-5} + \frac{10}{2^k}) \geq 3/4$[4].

When the samples are drawn from D_0, the probability that P outputs 1 is equal to the probability that in at least one of the t iterations, the randomly chosen non-zero vector v was orthogonal to the $2k$ samples drawn uniformly from $\{0,1\}^n$. Here, the probability is over v and the samples. By union bound, we can bound the probability of outputting 1 (error) by

$$10 \cdot 2^k \cdot \left(\frac{1}{2}\right)^{2k} = \frac{10}{2^k}.$$

For a fixed subspace $S \in L(k, n)$, the probability that we pick a non-zero vector $v \in \{0,1\}^n$ that is orthogonal to S is at least $\frac{2^{n-k}-1}{2^n-1} > 2^{-(k+1)}$. Therefore, when the samples are drawn from $D_1(S)$, the probability that P outputs 0 (error) is upper bounded by $\left(1 - \frac{1}{2^{k+1}}\right)^{10 \cdot 2^k} \leq e^{-5}$. Here, the probability is over the vectors v and the samples. Now to construct a constant width but $20k \cdot 2^k$ length branching program that distinguishes with probability at least $3/4$, we consider a bunch of branching programs each indexed by t vectors that are used in step 2 of the algorithm P. It's easy to see that for a fixed set of t vectors, P can be implemented by a constant width branching program. As, when the t vectors are uniformly distributed over $\{0,1\}^n$ (non-zero), P can distinguish with probability at least $3/4$ for every subspace $S \in L(k, n)$, there exists a fixing to the t vectors such that the corresponding branching program distinguishes between D_0 and $D_1(S)$ (when S is chosen uniformly at random from $L(k, n)$) with probability at least $3/4$.

[4] $k \geq 5$

Distinguishing Satisfiable Sparse Equations from Uniform

Informally, we study the problem of distinguishing between the cases when the samples are drawn from satisfiable sparse equations over $\mathbb{F}_2$ and when the samples are drawn from random sparse equations.

Formally, we consider the distinguishing problem between the following two distributions:

1. D_0: Distribution on $(n+1)$-length vectors $(v^1, v^2, ..., v^n, b) \in \{0,1\}^{n+1}$ where $\forall i \in [n]$, v^i is 1 with probability $\frac{k}{n}$ and 0 otherwise, and b is 1 with probability $\frac{1}{2}$ and 0 otherwise.
2. $D_1(x)$, $x \in \{0,1\}^n$: Distribution on $(n+1)$-length vectors $(v^1, v^2, ..., v^n, b) \in \{0,1\}^{n+1}$ where $\forall i \in [n]$, v^i is 1 with probability $\frac{k}{n}$ and 0 otherwise, and $b = \langle v, x \rangle$ where $v = (v^1, v^2, ..., v^n)$.

Here, k is the sparsity parameter.

We say that a distinguisher can distinguish satisfiable sparse equations of sparsity k from random ones if, when x is unknown and chosen uniformly at random from $\{0,1\}^n$, it can distinguish between D_0 and $D_1(x)$. Formally, a distinguisher L, after seeing m, has a success probability of p if

$$\frac{\Pr_{u_1,...,u_m \sim D_0}[L(u_1,...,u_m) = 0] + \Pr_{x \in_R \{0,1\}^n; u_1,...,u_m \sim D_1(x)}[L(u_1,...,u_m) = 1]}{2} \geq p \quad (2)$$

▶ **Theorem 14.** *For large enough n and $k > c_5 \log n$ (where c_5 is a large enough constant) and $k \leq \frac{n}{4}$, any algorithm that can distinguish random sparse parities of sparsity k on n variables from satisfiable ones, with success probability at least $\frac{1}{2} + 2^{-o(k)}$, requires either a memory of size $\Omega(nk)$ or $2^{\Omega(k)}$ samples.*

▶ Remark 15 (Tightness of our Lower Bound). We note two easy upper bounds that show that our results in Theorem 14 are almost tight. Firstly, we observe that there's an algorithm B_1 of memory $O(nk \log n)$ that uses $O(n)$ samples and can distinguish random sparse parities of sparsity k from satisfiable ones, with probability of at least $3/4$. B_1 just stores $O(n)$ samples (in $O(nk \log n)$ memory); if there exists x that satisfies all the samples, it outputs 1, otherwise it outputs 0. When the samples are satisfiable, that is, drawn from $D_1(x)$ (for some x), B_1 always outputs 1. When the samples are random, using the union bound, it's easy to see that the probability that there exists an x that satisfies all the $O(n)$ samples is exponentially small in n.

Second, there's an algorithm B_2 of constant memory that uses $O(n \cdot 2^{O(k)})$ samples and can distinguish random sparse parities of sparsity k from satisfiable ones, with probability of at least $3/4$. The probability that a learning algorithm sees sample (a, b), such that $a = (1, 0, ..., 0)$, is at least $\frac{ke^{-2k}}{n}$ for $k < n/2$; thus, one can just wait for say 5 such samples and see if the values of b are drawn randomly or are fixed, giving a constant memory and $O(ne^{2k})$ samples algorithm that distinguishes with high probability.

Refer to the full version of the paper [28] for the proof of Theorem 14. Briefly, we prove that using such a distinguisher, we can construct a branching program that learns an unknown bit vector x from sparse linear equations of sparsity k over $\mathbb{F}_2$. Unlike before, when we were able to lift, we are not able to directly lift the time-space tradeoffs of Theorem 8, because these lower bounds hold when the equations are of sparsity *exactly-k*. Following the proof of Theorem 8 in [29] very closely, we can prove the following lemma:

▶ **Lemma 16.** *Any branching program that learns $x \in \{0,1\}^n$ from random linear equations over $\mathbb{F}_2$ of type $(a, b) = (a^1, a^2, ..., a^n, b) \in \{0,1\}^{n+1}$, where $\forall i \in [n]$, $a^i = 1$ with probability $\frac{k}{n}$ and $b = \langle a, x \rangle$, with success probability 2^{-ck}, requires either width of size $2^{\Omega(n \cdot k)}$ or length of $2^{\Omega(k)}$ (where c is a small enough constant, $k \leq \frac{n}{4}$).*

The proof is given in full version [28]. Now through reduction, we are able to lift the time-space tradeoffs of Lemma 16 to get Theorem 14.

5 Sample-Memory Tradeoffs for Resilient Local PRGs

In this section, we prove our lower bound against memory bounded algorithms for distinguishing between streaming outputs of Goldreich's pseudorandom generator and perfectly random bits. Before stating our result in detail, we set up some notation and definitions that will be convenient for us in this section.

5.1 Formal Setup

A k-ary *predicate* P is a Boolean function $P : \{0,1\}^k \to \{0,1\}$. Let $\sum_{\alpha \subseteq [k]} \hat{P}(\alpha)\chi_\alpha$ be the Fourier polynomial for $(-1)^P$ ($(-1)^P(x) = (-1)^{P(x)}$). P is said to be *t-resilient* if t is the maximum positive integer such that $\hat{P}(\alpha) = 0$ whenever $|\alpha| < t$. In particular, the parity function $\langle \alpha, x \rangle$ is $|\alpha|$-resilient. Here, $\chi_\alpha : \{0,1\}^k \to \{-1,1\}$ is such that $\chi_\alpha(x) = (-1)^{\langle \alpha, x \rangle}$.

Let $[n]^{(k)}$ denote the set of all ordered k-tuples of exactly k elements of $[n]$. That is, no element of $[n]$ occurs more than once in any tuple of $[n]^{(k)}$. For any $a \in [n]^{(k)}$, let $a^i \in [n]$ denote the element of $[n]$ appearing in the ith position in a. Given $x \in \{0,1\}^n$ and $a \in [n]^{(k)}$, let $x^a \in \{0,1\}^k$ be defined so that $(x^a)^i = x^{a^i}$ for every $1 \le i \le k$.

For any k-ary predicate P, consider the problem of distinguishing between the following two distributions on $(a,b) \in [n]^{(k)} \times \{0,1\}$ where (a,b) are sampled as follows:

1. D_{null}: 1) Choose a uniformly at random from $[n]^{(k)}$, and 2) choose b uniformly at random and independently from $\{0,1\}$.
2. $D_{planted}(x)$, $x \in \{0,1\}^n$: 1) Choose a uniformly at random from $[n]^{(k)}$, and 2) set $b = P(x^a)$.

Note that a is chosen uniformly at random from $[n]^{(k)}$ in both distributions. However, while the bit b is independent of a in D_{null}, it may be correlated with a in $D_{planted}$.

A distinguisher for the above problem gets access to m i.i.d. samples $u_t = (a_t, b_t), t \in [m]$ from one of D_{null} and $D_{planted}(x)$ for a uniformly randomly chosen $x \in \{0,1\}^n$ and outputs either "planted" or "null". We say that the distinguisher succeeds with probability p if

$$\Pr_{u_1,\ldots,u_m \sim D_{null}}[L(u_1,\ldots,u_m) = \text{"null"}] + \Pr_{\substack{x \in_R \{0,1\}^n; \\ u_1,\ldots,u_m \sim D_{planted}(x)}}[L(u_1,\ldots,u_m) = \text{"planted"}]$$

is greater than $2p$. In the language used in the previous sections, think of "null" as being equivalent to 0 and "planted" being equivalent to 1, that is, $D_{null} \equiv D_0$ and $D_{planted}(x) \equiv D_1(x)$. Therefore, the success probability of the distinguisher L can be written as

$$\frac{\Pr_{u_1,\ldots,u_m \sim D_0}[L(u_1,\ldots,u_m) = 0] + \Pr_{x \in_R \{0,1\}^n; u_1,\ldots,u_m \sim D_1(x)}[L(u_1,\ldots,u_m) = 1]}{2} \ge p \quad (3)$$

In particular, if $x \in \{0,1\}^n$ is "revealed" to a distinguishing algorithm, then it is easy to use $\Theta(\log(1/\epsilon))$ samples and constant width branching program to distinguish correctly with probability at least $1 - \epsilon$ between D_{null} and $D_{planted}$.

5.2 Main Result

The main result of this section is the following sample-memory trade-off for any distinguisher.

▶ **Theorem 17.** *Let P be a t-resilient k-ary predicate. Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and $k < n/c$. Suppose there's an algorithm that distinguishes between D_{null} and $D_{planted}$ with probability at least $1/2 + s$ and uses $< n^\epsilon$ memory. Then, whenever $0 < t \le k < n^{\frac{(1-\epsilon)}{6}}/3$ and $s > c_1(\frac{n}{t})^{-(\frac{1-\epsilon}{36})t}$, the algorithm requires $(\frac{n}{t})^{(\frac{1-\epsilon}{36})t}$ samples. Here, c and c_1 are large enough constants.*

Note that when k is a constant, this theorem gives a sample-memory tradeoff even for $\Omega(n)$ memory.

Our argument yields a slightly better quantitative lower bound for the special case when P is the parity function, that is, $P(x) = (\sum_{i=1}^{k} x^i) \mod 2$. We will represent this function by Xor.

▶ **Theorem 18.** *Let $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$ and P be the parity predicate Xor on $k = t$ bits. Suppose there's an algorithm that distinguishes between D_{null} and $D_{planted}$ with probability at least $1/2 + s$ and uses $< n^{\epsilon}$ memory. Then for $k \leq n/c^5$, if $s > 3(\frac{n}{k})^{-(\frac{1-\epsilon}{18})k}$, the algorithm requires $(\frac{n}{k})^{(\frac{1-\epsilon}{18})k}$ samples.*

We prove both Theorem 17 and 18 via the same sequence of steps except for a certain quantitative bound presented in Lemma 19. In the next subsection, we give an algorithm that takes $\widetilde{O}(n^{\epsilon} + k)k$ memory and $\widetilde{O}(n^{(1-\epsilon)k})$ samples, and distinguishes between D_{null} and $D_{planted}$ for any predicate P, with probability 99/100. Thus, the lower bounds are almost tight up to constant factors in the exponent for the parity predicate. The question of whether there exists an algorithm that runs in $O(n^{(1-\epsilon)t})$ samples and $O(n^{\epsilon})$ memory, and distinguishes between D_{null} and $D_{planted}$ with high probability, for t-resilient predicates P, remains open.

5.3 Tightness of the Lower Bound

In this section, we observe that there exists an algorithm A that takes $O((n^{\epsilon} + k) \cdot k \log n)$ memory and $O(n^{(1-\epsilon)k} \cdot (n^{\epsilon} + k))$ samples, and distinguishes between D_{null} and $D_{planted}$ for any predicate P, with probability 99/100 (for $n^{\epsilon} > 10$).

A runs over $4n^{(1-\epsilon)k} \cdot (n^{\epsilon} + k)$ samples and stores the first $2(n^{\epsilon} + k)$ samples $(a, b) \in [n]^{(k)} \times \{0, 1\}$ such that $a^i \leq n^{\epsilon} + k, \forall i \in [k]$, that is, the bit b depends only on the first $n^{\epsilon} + k$ bits of x under the distribution $D_{planted}(x)$. If A encounters less than $2(n^{\epsilon} + k)$ samples of the above mentioned form, A outputs 1 ("planted"). Otherwise, A goes over all the possibilities of the first $n^{\epsilon} + k$ bits of x ($2^{n^{\epsilon}+k}$ possibilities in total) and checks if it could have generated the stored samples. If there exists a $y \in \{0, 1\}^{n^{\epsilon}+k}$ that generated the stored samples, A outputs 1 ("planted"), otherwise A outputs 0.

It's easy to see that A uses $m = 4n^{(1-\epsilon)k} \cdot (n^{\epsilon} + k)$ samples and at most $2(n^{\epsilon} + k) \cdot k \log n$ memory (as it takes only $k \log n$ memory to store a sample). Next, we calculate the probability of success. Let Z_j be a random variable as follows: $Z_j = 1$ if the j^{th} sample (a_j, b_j) is such that $a_j^i \leq n^{\epsilon} + k, \forall i \in [k]$ and 0 otherwise.

$$\Pr[Z_j = 1] = \frac{|[n^{\epsilon} + k]^{(k)}|}{|[n]^{(k)}|} \geq n^{-(1-\epsilon)k}$$

And $\mathbf{E}[\sum_{j=1}^{m} Z_j] = 4(n^{\epsilon} + k)$. By Chernoff bound, $\Pr[\sum_j Z_j < 2(n^{\epsilon} + k)] \leq e^{-\frac{4(n^{\epsilon}+k)}{8}} \leq \frac{1}{100}$. Therefore, the probability that A stores $2(n^{\epsilon} + k)$ samples is at least 99/100. It's easy to see that A always outputs 1 when the samples are generated from $D_{planted}(x)$ for any x.

The probability that A outputs 1, given that it stored $2(n^{\epsilon}+k)$ samples, when the samples are generate from D_{null} is equal to the probability that there exists a $y \in \{0, 1\}^{n^{\epsilon}+k}$ that could have generated the stored samples. Let $(a'_1, b'_1), ..., (a'_{2(n^{\epsilon}+k)}, b'_{2(n^{\epsilon}+k)})$ be the stored samples. There are at most $2^{n^{\epsilon}+k}$ sequences of $b'_1, ..., b'_{2(n^{\epsilon}+k)}$ generated by some y given

[5] c is a large enough constant

$\{a'_j\}_{j\in[2(n^\epsilon+k)]}$. As, under D_{null}, b is chosen uniformly at random from $\{0,1\}$, the probability that there exists $y \in \{0,1\}^{n^\epsilon+k}$ that could have generated the stored samples is at most $\frac{2^{n^\epsilon+k}}{2^{2(n^\epsilon+k)}} = 2^{-(n^\epsilon+k)} \leq 1/100$. Hence, the probability of success is at least 99/100.

5.4 Proof of Theorems 17 and 18

For the full proof, please refer to the full version of the paper [28]. In this section, we give a very brief outline of the proof and state a crucial lemma. Fix a t-resilient k-ary predicate P. Let B be a branching program with bounded width and length, that distinguishes between D_{null} and $D_{planted}(x)$ (x is uniformly distributed over $\{0,1\}^n$) for the predicate P, with non-negligible probability.

We first use hybrid argument to obtain that the branching program B must have a non-trivial probability of distinguishing with a single sample. Towards this, define $H_j(x)$ to be the distribution over m samples where the first j samples are drawn from $D_{planted}(x)$ and the remaining $m-j$ samples are from D_{null}. Using hybrid argument, there is a $j' \in \{1,...,m\}$ such that B distinguishes between the hybrids $H_{j'-1}(x)$ and $H_{j'}(x)$ with non-negligible probability. Then to contradict, we show that a bounded-width and bounded-length B cannot distinguish between the hybrids $H_{j'-1}(x)$ and $H_{j'}(x)$.

Let v_1 be a vertex of the branching program B after seeing the first $j'-1$ (generated from a pseudorandom source with a seed x picked uniformly at random from $\{0,1\}^n$). Let $\mathbb{P}_{x|v_1}$ be the probability distribution of the random variable x conditioned on reaching v_1. If the j'th sample is generated using the same pseudorandom source (as in $H_{j'}(x)$), then $\forall a \in [n]^{(k)}$, the bit b is 0 with probability $\sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x')$ and 1 with probability $1 - \sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x')$. If the j'th sample is generated using the uniform source (as in $H_{j'-1}(x)$), then $\forall a \in [n]^{(k)}$, the bit b is 0 with probability 1/2 and 1 with probability 1/2. Thus, for any a, B can distinguish with at most $\left|\sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x') - 1/2\right|$ advantage. We show that when B has bounded width, then with high probability, it reaches a vertex v_1 such that $\mathbb{P}_{x|v_1}$ has high min-entropy. We then use t-resiliency of P to show that when $\mathbb{P}_{x|v_1}$ has high min-entropy, then with high probability over $a \in [n]^{(k)}$, the expression $\left|\frac{1}{2} - \sum_{x':P(x'^a)=0} \mathbb{P}_{x|v_1}(x')\right|$ is small (the following lemma). Hence, with high probability, it's hard for B to distinguish whether the j'th sample was generated from $H_{j'}(x)$ or $H_{j'-1}(x)$. Note that the last $m-j$ samples, generated by a uniform source, can't better this advantage.

Define $T_l = \{\bar{a} \in \{0,1\}^n : \sum_{i=1}^n \bar{a}^i = l\}$ for $l \in \mathbb{N}$. For $\epsilon > 0, k \geq t > 0$, let L be the set of vertices v such that $\mathbb{P}_{x|v}$ has min-entropy of at least $(n - n^\epsilon - t\log(n/t))$, that is, $\forall x' \in \{0,1\}^n$, $\mathbb{P}_{x|v}(x') \leq 2^{n^\epsilon - n} \cdot (n/t)^t$.

▶ **Lemma 19.** *For all $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$, $0 < t \leq k < \{\frac{n}{c}, n^{\frac{(1-\epsilon)}{6}}/3\}$, and $v_1 \in L$,*

$$\left|\sum_{x'} \mathbb{P}_{x|v_1}(x') \cdot (-1)^{P(x'^a)}\right| \leq c_2 n^{-(\frac{1-\epsilon}{18})t}$$

for all but at most $c_2 n^{-(\frac{1-\epsilon}{18})t}$ fraction of $a \in [n]^{(k)}$ (recall that P is a t-resilient k-ary predicate).

For all $v_1 \in L$, $0 < \epsilon < 1 - 3\frac{\log 24}{\log n}$, $0 < k < \frac{n}{c}$,

$$\left|\sum_{x'} \mathbb{P}_{x|v_1}(x') \cdot (-1)^{Xor(x'^a)}\right| \leq 2\left(\frac{n}{k}\right)^{-(\frac{1-\epsilon}{9})k}$$

for all but at most $2(\frac{n}{k})^{-(\frac{1-\epsilon}{9})k}$ fraction of $a \in [n]^{(k)}$.

Here, c and c_2 are large enough constants.

Please refer to the full version [28] for the proof of the above lemma.

References

1. Michael Alekhnovich, Eli Ben-Sasson, Alexander A. Razborov, and Avi Wigderson. Pseudorandom generators in propositional proof complexity. *SIAM J. Comput.*, 34(1):67–88, 2004. doi:10.1137/S0097539701389944.
2. Sarah R. Allen, Ryan O'Donnell, and David Witmer. How to refute a random CSP. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*, pages 689–708. IEEE Computer Soc., Los Alamitos, CA, 2015.
3. Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. Cryptology ePrint Archive, Report 2015/730, 2015. URL: https://eprint.iacr.org/2015/730.
4. Benny Applebaum. Pseudorandom generators with long stretch and low locality from random local one-way functions. *SIAM J. Comput.*, 42(5):2008–2037, 2013.
5. Benny Applebaum. Cryptographic hardness of random local functions - survey. *Computational Complexity*, 25(3):667–722, 2016.
6. Benny Applebaum, Boaz Barak, and Avi Wigderson. Public-key cryptography from different assumptions. In *STOC'10—Proceedings of the 2010 ACM International Symposium on Theory of Computing*, pages 171–180. ACM, New York, 2010.
7. Benny Applebaum, Andrej Bogdanov, and Alon Rosen. A dichotomy for local small-bias generators. *J. Cryptology*, 29(3):577–596, 2016.
8. Benny Applebaum and Shachar Lovett. Algebraic attacks against random local functions and their countermeasures. In *STOC*, pages 1087–1100. ACM, 2016.
9. Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Trans. Information Theory*, 48(6):1668–1680, 2002. doi:10.1109/TIT.2002.1003845.
10. Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, pages 65–79, 1999. doi:10.1007/3-540-48405-1_5.
11. Boaz Barak, Zvika Brakerski, Ilan Komargodski, and Pravesh K. Kothari. Limits on low-degree pseudorandom generators (or: Sum-of-squares meets program obfuscation). In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 649–679, 2018. doi:10.1007/978-3-319-78375-8_21.
12. Boaz Barak, Siu On Chan, and Pravesh K. Kothari. Sum of squares lower bounds from pairwise independence [extended abstract]. In *STOC'15—Proceedings of the 2015 ACM Symposium on Theory of Computing*, pages 97–106. ACM, New York, 2015.
13. Boaz Barak and Ankur Moitra. Noisy tensor completion via the sum-of-squares hierarchy. In *COLT*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 417–445. JMLR.org, 2016.
14. Paul Beame, Shayan Oveis Gharan, and Xin Yang. Time-space tradeoffs for learning finite functions from random evaluations, with applications to polynomials. In *Conference On Learning Theory*, pages 843–856, 2018.
15. Christian Cachin and Ueli M. Maurer. Unconditional security against memory-bounded adversaries. In *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, pages 292–306, 1997. doi:10.1007/BFb0052243.
16. Michael S. Crouch, Andrew McGregor, Gregory Valiant, and David P. Woodruff. Stochastic streams: Sample complexity vs. space complexity. In *ESA*, volume 57 of *LIPIcs*, pages 32:1–32:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

17 Mary Cryan and Peter Bro Miltersen. On pseudorandom generators in NC. In *26th International Symposium on Mathematical Foundations of Computer Science, MFCS*, pages 272–284, 2001.

18 Yuval Dagan and Ohad Shamir. Detecting correlations with little memory and communication. In *Conference On Learning Theory*, pages 1145–1198, 2018.

19 Wei Dai, Stefano Tessaro, and Xihu Zhang. Super-linear time-memory trade-offs for symmetric encryption. Cryptology ePrint Archive, Report 2020/663, 2020. URL: `https://eprint.iacr.org/2020/663`.

20 Amit Daniely. Complexity theoretic limitations on learning halfspaces. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 105–117. ACM, 2016. `doi:10.1145/2897518.2897520`.

21 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. More data speeds up training time in learning halfspaces over sparse vectors. In *NIPS*, pages 145–153, 2013.

22 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. The complexity of learning halfspaces using generalized linear methods. In *COLT*, volume 35 of *JMLR Workshop and Conference Proceedings*, pages 244–286. JMLR.org, 2014.

23 Amit Daniely, Nati Linial, and Shai Shalev-Shwartz. From average case complexity to improper learning complexity. In *STOC*, pages 441–448. ACM, 2014.

24 Amit Daniely and Shai Shalev-Shwartz. Complexity theoretic limitations on learning dnf's. In Vitaly Feldman, Alexander Rakhlin, and Ohad Shamir, editors, *Proceedings of the 29th Conference on Learning Theory, COLT 2016, New York, USA, June 23-26, 2016*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 815–830. JMLR.org, 2016. URL: `http://proceedings.mlr.press/v49/daniely16.html`.

25 Ilias Diakonikolas, Themis Gouleakis, Daniel M Kane, and Sankeerth Rao. Communication and memory efficient testing of discrete distributions. In *Conference on Learning Theory*, pages 1070–1106, 2019.

26 Stefan Dziembowski and Ueli M. Maurer. On generating the initial key in the bounded-storage model. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 126–137, 2004. `doi:10.1007/978-3-540-24676-3_8`.

27 Uriel Feige. Relations between average case complexity and approximation complexity. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, pages 534–543. ACM, New York, 2002. `doi:10.1145/509907.509985`.

28 Sumegha Garg, Pravesh K Kothari, and Ran Raz. Time-space tradeoffs for distinguishing distributions and applications to security of goldreich's prg. *arXiv preprint*, 2020. `arXiv:2002.07235`.

29 Sumegha Garg, Ran Raz, and Avishay Tal. Extractor-based time-space lower bounds for learning. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 990–1002. ACM, 2018.

30 Sumegha Garg, Ran Raz, and Avishay Tal. Time-space lower bounds for two-pass learning. In *34th Computational Complexity Conference (CCC 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

31 Oded Goldreich. Candidate one-way functions based on expander graphs. *Electronic Colloquium on Computational Complexity (ECCC)*, 7(90), 2000.

32 Jiaxin Guan and Mark Zhandary. Simple schemes in the bounded storage model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 500–524. Springer, 2019.

33 Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, Manoj Prabhakaran, and Amit Sahai. Efficient non-interactive secure computation. In *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 406–425, 2011. `doi:10.1007/978-3-642-20465-4_23`.

34 Joseph Jaeger and Stefano Tessaro. Tight time-memory trade-offs for symmetric encryption. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 467–497. Springer, 2019.

35 Gillat Kol, Ran Raz, and Avishay Tal. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1067–1080. ACM, 2017.

36 Pravesh K. Kothari and Roi Livni. Improper learning by refuting. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 55:1–55:10, 2018. doi:10.4230/LIPIcs.ITCS.2018.55.

37 Pravesh K. Kothari, Ryuhei Mori, Ryan O'Donnell, and David Witmer. Sum of squares lower bounds for refuting any CSP. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 132–145, 2017. doi:10.1145/3055399.3055485.

38 Huijia Lin. Indistinguishability obfuscation from constant-degree graded encoding schemes. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I*, pages 28–57, 2016. doi:10.1007/978-3-662-49890-3_2.

39 Huijia Lin. Indistinguishability obfuscation from sxdh on 5-linear maps and locality-5 prgs. Cryptology ePrint Archive, Report 2016/1096, 2016. URL: https://eprint.iacr.org/2016/1096.

40 Huijia Lin and Stefano Tessaro. Indistinguishability obfuscation from trilinear maps and block-wise local prgs. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 630–660, 2017. doi:10.1007/978-3-319-63688-7_21.

41 Huijia Lin and Vinod Vaikuntanathan. Indistinguishability obfuscation from ddh-like assumptions on constant-degree graded encodings. 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 11–20. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.11.

42 Alex Lombardi and Vinod Vaikuntanathan. Limits on the locality of pseudorandom generators and applications to indistinguishability obfuscation. In *Theory of Cryptography - 15th International Conference, TCC*, volume 10677, pages 119–137. Springer, 2017.

43 Alex Lombardi and Vinod Vaikuntanathan. Minimizing the complexity of Goldreich's pseudorandom generator. *IACR Cryptology ePrint Archive*, page 277, 2017.

44 Ueli M. Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *J. Cryptology*, 5(1):53–66, 1992. doi:10.1007/BF00191321.

45 Dana Moshkovitz and Michal Moshkovitz. Mixing implies lower bounds for space bounded learning. In *COLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 1516–1566. PMLR, 2017.

46 Dana Moshkovitz and Michal Moshkovitz. Entropy samplers and strong generic lower bounds for space bounded learning. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

47 Michal Moshkovitz and Naftali Tishby. Mixing complexity and its applications to neural networks. *arXiv preprint*, 2017. arXiv:1703.00729.

48 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On e-biased generators in NC0. In *FOCS*, pages 136–145. IEEE Computer Society, 2003.

49 Elchanan Mossel, Amir Shpilka, and Luca Trevisan. On ϵ-biased generators in NC^0. *Random Structures Algorithms*, 29(1):56–81, 2006. doi:10.1002/rsa.20112.

50 N. Nisan and D. Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–52, 1996.

51 Ryan O'Donnell and David Witmer. Goldreich's PRG: evidence for near-optimal polynomial stretch. In *IEEE 29th Conference on Computational Complexity—CCC 2014*, pages 1–12. IEEE Computer Soc., Los Alamitos, CA, 2014. doi:10.1109/CCC.2014.9.

52 Prasad Raghavendra, Satish Rao, and Tselil Schramm. Strongly refuting random csps below the spectral threshold. In *STOC*, pages 121–131. ACM, 2017.

53 Ran Raz. Fast learning requires good memory: A time-space lower bound for parity learning. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 266–275. IEEE, 2016.

54 Ran Raz. A time-space lower bound for a large class of learning problems. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 732–742, 2017.

55 Ohad Shamir. Fundamental limits of online and distributed algorithms for statistical learning and estimation. In Zoubin Ghahramani, Max Welling, Corinna Cortes, Neil D. Lawrence, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 163–171, 2014. URL: http://papers.nips.cc/book/advances-in-neural-information-processing-systems-27-2014.

56 Vatsal Sharan, Aaron Sidford, and Gregory Valiant. Memory-sample tradeoffs for linear regression with small error. *arXiv preprint*, 2019. arXiv:1904.08544.

57 Jacob Steinhardt, Gregory Valiant, and Stefan Wager. Memory, communication, and statistical queries. In *COLT*, volume 49 of *JMLR Workshop and Conference Proceedings*, pages 1490–1516. JMLR.org, 2016.

58 Stefano Tessaro and Aishwarya Thiruvengadam. Provable time-memory trade-offs: symmetric cryptography against memory-bounded adversaries. In *Theory of Cryptography Conference*, pages 3–32. Springer, 2018.

59 Salil P. Vadhan. On constructing locally computable extractors and cryptosystems in the bounded storage model. In *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 61–77. Springer, 2003.

60 Salil P. Vadhan. On learning vs. refutation. In *COLT*, volume 65 of *Proceedings of Machine Learning Research*, pages 1835–1848. PMLR, 2017.

61 Gregory Valiant and Paul Valiant. Information theoretically secure databases. *arXiv preprint*, 2016. arXiv:1605.02646.

Streaming Verification for Graph Problems: Optimal Tradeoffs and Nonlinear Sketches



Amit Chakrabarti
Dartmouth College, Hanover, NH, USA

Prantar Ghosh
Dartmouth College, Hanover, NH, USA

Justin Thaler
Georgetown University, Washington, DC, USA

Abstract

We study graph computations in an enhanced data streaming setting, where a space-bounded client reading the edge stream of a massive graph may delegate some of its work to a cloud service. We seek algorithms that allow the client to *verify* a purported *proof* sent by the cloud service that the work done in the cloud is correct. A line of work starting with Chakrabarti et al. (ICALP 2009) has provided such algorithms, which we call *schemes*, for several statistical and graph-theoretic problems, many of which exhibit a tradeoff between the length of the proof and the space used by the streaming verifier.

This work designs new schemes for a number of basic graph problems – including triangle counting, maximum matching, topological sorting, and single-source shortest paths – where past work had either failed to obtain smooth tradeoffs between these two key complexity measures or only obtained suboptimal tradeoffs. Our key innovation is having the verifier compute certain *nonlinear* sketches of the input stream, leading to either new or improved tradeoffs. In many cases, our schemes in fact provide optimal tradeoffs up to logarithmic factors.

Specifically, for most graph problems that we study, it is known that the product of the verifier's space cost v and the proof length h must be at least $\Omega(n^2)$ for n-vertex graphs. However, matching upper bounds are only known for a handful of settings of h and v on the curve $h \cdot v = \tilde{\Theta}(n^2)$. For example, for counting triangles and maximum matching, schemes with costs lying on this curve are only known for $(h = \tilde{O}(n^2), v = \tilde{O}(1))$, $(h = \tilde{O}(n), v = \tilde{O}(n))$, and the trivial $(h = \tilde{O}(1), v = \tilde{O}(n^2))$. A major message of this work is that by exploiting nonlinear sketches, a significant "portion" of costs on the tradeoff curve $h \cdot v = n^2$ can be achieved.

2012 ACM Subject Classification Theory of computation → Streaming models; Theory of computation → Interactive proof systems; Computer systems organization → Cloud computing

Keywords and phrases data streams, interactive proofs, Arthur-Merlin, graph algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.22

Category RANDOM

Related Version A full version of the paper is available at `https://arxiv.org/pdf/2007.03039.pdf`.

Funding *Amit Chakrabarti*: Work supported in part by NSF under award CCF-1907738.
Prantar Ghosh: Work supported in part by NSF under award CCF-1907738.
Justin Thaler: Work supported by NSF SPX award CCF-1918989, and NSF CAREER award CCF-1845125. Parts of this work were performed while visiting the Simons Institute for the Theory of Computing.


© Amit Chakrabarti, Prantar Ghosh, and Justin Thaler;
licensed under Creative Commons License CC-BY

1 Introduction

It is far easier to verify a proof than to find one. This intuitively clear fact has been given precise meanings in several settings , leading to such landmark results as the IP = PSPACE [36] and PCP Theorems [3, 4]. There is a growing body of work on results of this flavor for space-efficient computations on large data streams [35]. In this setting, a space-bounded client (henceforth named Verifier) that can only process inputs in the restrictive *data streaming* setting has access to a computationally powerful entity (henceforth named Prover), such as cloud computing service, that has no such space limitations. As past work has shown, many fundamental problems that are intractable in the plain data-streaming model – in the sense that they cannot be solved using sublinear space – do admit nontrivial solutions in this Verifier/Prover model, *without* Verifier having to trust Prover blindly.

An algorithm in this model specifies a protocol to be followed by Verifier and Prover so that the former may compute some function $f(\sigma)$ of the input stream σ. Prover, by performing the specified actions honestly, convinces Verifier to output the correct value $f(\sigma)$. However, if Prover fails to follow the protocol, whether out of malice or error (modeling hardware, software, or network faults in the cloud service), then Verifier is highly likely to detect this and *reject*. Past work has considered a few different instances of this setup, such as (a) *annotated* data streaming algorithms [12, 30] – also called *online schemes* – where the parties read σ together and the protocol consists of Prover streaming a "help message" (a.k.a. proof) to Verifier either during stream processing and/or at the end; (b) *prescient schemes* [11], which are a variant of the above where Prover knows all of σ before Verifier sees it; (c) *streaming interactive proofs* (SIPs) [13, 19], where Verifier and Prover engage in multiple rounds of communication.

This work focuses on the first and arguably best-motivated of these models, namely, online schemes. We simply call them *schemes*. We give new and improved schemes for several graph-theoretic problems, including triangle counting, maximum matching, topological sorting, and shortest paths. In all cases, the input is a huge n-vertex graph G given as a stream σ of edge insertions and/or deletions. While most of our problems have been studied before, we give schemes that (a) have better complexity parameters, in some cases achieving optimality, and (b) use cleverer algebraic encodings of the relevant combinatorial problems, often exploiting the ability of a streaming algorithm to compute *nonlinear* sketches.

1.1 Setup, Terminology, and Motivation

We formalize the setup described above. A scheme for a function f specifies three things: (i) a space-bounded data streaming algorithm used by Verifier to process the input σ and compute a summary $\mathcal{V}_R(\sigma)$, using random coins R; (ii) a help function used by Prover to send a message $\mathcal{H}(\sigma)$ to Verifier as a "proof stream" after the input stream ends;[1] and (iii) an output algorithm $\text{out}_R(\mathcal{V}_R(\sigma), \mathcal{H}(\sigma))$ capturing Verifier's work during and after the proof stream, which produces values in $\text{range}(f) \cup \{\perp\}$, where an output of $\perp$ indicates "reject." If $\mathcal{V}_R$ and out_R run in $O(v)$ bits of space and $\mathcal{H}$ provides $O(h)$ bits of help, then this scheme is called an (h, v)-*scheme*. A scheme is interesting if we can use $h > 0$ to achieve a value of v asymptotically smaller than what is feasible or known for a basic streaming algorithm, where $h = 0$. A scheme is said to have

- completeness error ε_c if $\forall \sigma\, \exists \mathcal{H} : \Pr_R[\text{out}_R(\mathcal{V}_R(\sigma), \mathcal{H}(\sigma)) = f(\sigma)] \geqslant 1 - \varepsilon_c$;
- soundness error ε_s if $\forall \sigma, \mathcal{H}' : \Pr_R[\text{out}_R(\mathcal{V}_R(\sigma), \mathcal{H}'(\sigma)) \notin \{f(\sigma), \perp\}] \leqslant \varepsilon_s$.

[1] A more general (though seldom used) model allows Prover to send help messages after each data item in σ.

In designing schemes, we will aim for $\varepsilon_s \leqslant 1/3$, which can be reduced further via parallel repetition in standard ways. We will also achieve perfect completeness, i.e., $\varepsilon_c = 0$. For an (h, v)-scheme we refer to h as its hcost (short for "help cost") and v as its vcost ("verification cost"). We use the notation $[h, v]$-scheme as a shorthand for an $(\tilde{O}(h), \tilde{O}(v))$-scheme.[2]

It is intuitive that the parameters h and v are in tension, suggesting that they can be traded off against one another. Most of our algorithms do obtain such tradeoffs. We emphasize that actually obtaining a smooth tradeoff for large ranges of h and v values is not automatic: indeed, an important contribution of this work is to obtain such tradeoffs for problems where past work gave comparable results only for specific settings of h and v.

When studying the results discussed below, it is useful to keep a few cost regimes in mind. We focus on graph problems on n-vertex inputs. An (h, v)-scheme for such a problem is *sublinear* if $h = o(n^2)$ and $v = o(n^2)$; *frugal* if it is sublinear and achieves the stronger guarantee $v = o(n)$; and *laconic* if it is sublinear and achieves the stronger guarantee $h = o(n)$.

Many graph problems are *intractable* in the basic one-pass streaming model, meaning that they provably require $\Omega(n^2)$ space. Past work [12] implies that any (h, v)-scheme for such a problem must have $hv = \Omega(n^2)$. Thus, an $[h, v]$-scheme with $hv = O(n^2)$ for an intractable problem has achieved an optimal tradeoff, up to logarithmic factors. All of the problems we consider in this paper (except for counting connected components) are intractable for dense graphs (i.e., graphs with $\Omega(n^2)$ edges).

Frugal schemes are important when Verifier is so starved for space that it cannot afford to store even a constant fraction of the vertices. They are also very interesting from a theoretical standpoint, since even "easy" graph problems require at least $\Omega(n)$ space in the basic streaming model. On the other hand, laconic schemes are naturally motivated by settings where Verifier does not have streaming access to the proof and has to store it in full. Consider for example a retail client that uploads transactions to the cloud as they occur. It makes sense to have uploaded even terabytes of information *in total* over a long period of time: days, months, or years. However, it might not be reasonable for the cloud to transfer a proof consisting of, say, tens of gigabytes to the client. From a theoretical standpoint, in solving an intractable problem, if Verifier has to store the proof, there is no reason to ever try to reduce vcost to $o(n)$, since hcost will then blow up to $\omega(n)$.

1.2 Problems, Results, and Comparisons with Related Work

Throughout, the input graph G will be on the fixed vertex set $V = [n] := \{1, \ldots, n\}$ and will have m edges. Many results will be stated in terms of tunable parameters $t, s \in \mathbb{Z}^+$ that must satisfy $ts \geqslant n$. Since bounds are asymptotic, this condition can be read as $ts = n$.

Triangle Counting. Our starting point is the triangle counting problem (henceforth, TriangleCount), studied heavily in past work on graph streaming [7, 8, 10, 24, 25, 27, 34, 38]. Given a multigraph G as a dynamic stream (i.e., insertions and deletions), the goal is to compute T, the number of triangles in G. The exact counting version studied in this paper is an intractable problem in the sense of Section 1.1: it requires $\Omega(n^2)$ space in basic streaming.

As noted in Table 1, we give several new algorithms for TriangleCount. Our $[nt^2, s]$-scheme improves upon the best known frugal scheme for the problem: for a fixed hcost $\geqslant n$, it improves the vcost from $v^{4/3}$ to v, and for a fixed vcost $v \leqslant n$, it improves the hcost from $n^3/v^{3/2}$ to n^3/v^2. Our $[t, ns]$-scheme is not only the first laconic scheme for the problem but

[2] The notation $\tilde{O}(\cdot)$ hides factors polynomial in $\log n$.

Table 1 Summary of results on the problems considered in this paper. A scheme is deemed *optimal* if it has help cost at most h and space cost at most v for at least one pair h, v such that $h \cdot v \leqslant \tilde{O}(L)$, whereas it is known that *any* (h, v) scheme that applies to all graphs requires $h \cdot v \geqslant \Omega(L)$. A blank space in the *Tradeoff* column indicates that it remains open whether the scheme can be strictly improved. Here, α' is the size of a maximum matching in the input graph, K is the length of a shortest v_s–v_t path, D is the maximum distance from the source to the any other reachable vertex, and W is the maximum weight of an edge.

Problem	Scheme	Tradeoff	Reference
TriangleCount	$[h, v]$; $hv = n^3$	Suboptimal	[12]
	$[n^2, 1]$	Optimal	[12]
	$[n, n]$	Optimal	[38]
	$[t^3, s^2]$; $ts = n$	Suboptimal	[14]
	$[nt^2, s]$; $ts = n$		Theorem 4
	$[t, ns]$; $ts = n$	Optimal	Theorem 5
	$[mn/\sqrt{v},\ v]$	Suboptimal	[11]
	$[m + h, v]$; $hv = n^2$		Theorem 10
TriangleCount-Adj	$[h, v]$; $hv = n^2$		Theorem 11
MaxMatching	$[m, 1]$	Optimal	[18]
	$[n, n]$	Optimal	[38]
	$[t^3, s^2]$; $ts = n$	Suboptimal	[14]
	$[nt, s]$; $ts = n$	Optimal	Theorem 8
	$[\alpha' + h, v]$; $hv = n^2$		Theorem 9
MIS	$[nt, s]$; $ts = n$	Optimal	Theorem 12
Acyclicity/TopoSort	$[m, 1]$	Optimal	[18]
	$[nt, s]$; $ts = n$	Optimal	Theorem 13; Corollary 14
st-ShortestPath	$[Dnt, s]$; $ts = n$		[18]
	$[Kn, n]$		[14]
	$[Knt, s]$; $ts = n$		Corollary 18
Unweighted SSSP	$[Dnt, s]$; $ts = n$		Theorem 17
Weighted SSSP	$[m + n, 1]$	Optimal	[18]
	$[DWn, n]$		Theorem 19
	$[Dn, Wn]$		Theorem 20

also achieves smooth optimal tradeoff in its parameter range; thus, it settles the complexity of the problem in the laconic regime. The $[m+h, v]$-scheme whenever $hv = n^2$ generalizes the $[n^2, 1]$-scheme from prior work for any m-edge graph (for the setting $h = m$ and $v = n^2/m$) and is interesting in the frugal regime for sparse graphs.

The problem has also been studied in the adjacency-list model (call it TriangleCount-Adj) [10, 26, 31, 34], where the stream presents the full neighbor list for each vertex contiguously. We give an $[h, v]$-scheme for any $hv = n^2$ for TriangleCount-Adj (again, exact counting). In basic streaming, there is no nontrivial algorithm for computing T exactly, or even approximately when T is small; in fact, under a long-standing conjecture in communication complexity, these problems require $\Omega(m)$ space [26].

Maximum Matching. There is a recent and ongoing flurry of activity on streaming algorithms for MaxMatching, the problem of computing the cardinality $\alpha'(G)$ of a maximum-sized matching[3] in G [6, 16, 21, 22, 28, 33, 20, 29]. The exact version of the problem (which is what we study here) is intractable. For the special case of detecting whether a bipartite graph has a perfect matching, there is a frugal $[nt, s]$-scheme [12], which achieves optimal tradeoff. See Table 1 for previous results for the general problem.

In this work, we give (i) the first optimal frugal $[nt, s]$-scheme for the general MaxMatching problem, settling its complexity in the frugal regime, and (ii) an $[\alpha' + h, v]$-scheme whenever $hv = n^2$, which yields a laconic scheme provided $\alpha'(G) = o(n)$. Obtaining a fully general laconic scheme remains an interesting open problem and we suspect that it will require a breakthrough in exploiting the problem's combinatorial structure.

Further Graph Problems and a Common Framework. We obtain new schemes for the MIS problem, which asks for an inclusion-wise maximal independent set of vertices; the Acyclicity problem, which asks whether the input digraph is acyclic; and the TopoSort problem, which asks for a vertex ordering of the input DAG that orients all edges "forwards." In each case, we give an $[nt, s]$-scheme. Recent results show that MIS [5, 17] and TopoSort [15] are intractable in basic streaming, so our schemes are optimal in the frugal regime. Importantly, these schemes, the frugal MaxMatching scheme, and two of the TriangleCount schemes all fit a common framework: they boil the problem down to counting the number of edges in one or more induced subgraphs of the input graph. Our scheme for this InducedEdgeCount problem could be a useful technical result for future work.

Shortest Paths. The single-source shortest path (SSSP) problem is perhaps the most basic problem in classic graph algorithms. In the streaming setting, even the special case of undirected v_s–v_t connectivity in constant-diameter graphs is intractable [21]. As Table 1 shows, our $[Dnt, s]$-scheme for unweighted SSSP (where D is the maximum distance from the source vertex v_s to any vertex reachable from it) generalizes the result of Cormode et al. [18] from st-ShortestPath to SSSP. Again, as a corollary, we obtain a $[Knt, s]$-scheme for st-ShortestPath, where K is the length of a shortest v_s–v_t path. This result generalizes the $[Kn, n]$-scheme of Chakrabarti and Ghosh [14] and improves upon the $[Dnt, s]$-scheme of Cormode et al. [18], since K can be arbitrarily smaller than D. The schemes for the weighted version are interesting for small D and W, where W is the maximum weight of any edge.

1.3 Other Related Works

Abdullah et al. [1] studied the TriangleCount and MaxMatching problems in the stronger SIP model that allows rounds of interaction between Prover and Verifier. For TriangleCount, they gave a $(\log^2 n, \log^2 n)$-SIP using $\log n$ rounds of interaction. They also designed an $(n^{1/\gamma} \log n, \log n)$-SIP with $\gamma = O(1)$ rounds. For the weighted MaxMatching problem, they gave a $(\rho + n^{1/\gamma'} \log n, \log n)$-SIP using γ rounds of interaction, where γ' is a linear function of γ, and ρ is the weight of an optimal matching.

Early works on the concept of annotated streams include Tucker et al. [39] and Yi et al. [41], who studied *stream punctuations* and *stream outsourcing* respectively. Motivated by these works, Chakrabarti et al. [12] then formalised the model theoretically as the *annotated streaming model* and gave schemes for statistical streaming problems including

[3] The notation $\alpha'(G)$ is by analogy with $\alpha(G)$, which denotes the cardinality of a maximum independent set of *vertices*. It can be found, e.g., in the textbook by West [40].

frequency moments and heavy hitters, along with some basic results for graph problems. This non-interactive model was subsequently studied by multiple works including Klauck and Prakash [30], Cormode et al. [18], and Chakrabarti et al. [11]. Subsequent works considered generalized versions of the model, allowing rounds of interaction. These include *Arthur-Merlin streaming protocols* of Gur and Raz [23] and the *streaming interactive proofs* (SIP) of Cormode et al. [19]. Chakrabarti et al. [13] and Abdullah et al. [1] further studied this generalized setting. We refer to the expository article of Thaler [37] for a more detailed survey of this area.

1.4 Our Techniques

Sum-Check and Polynomial Encodings. As with much prior work in this area (and probabilistic proof systems more generally), our schemes are variants of the famous *sum-check protocol* of Lund et al. [32]. Specialized to our (non-interactive) schemes, this protocol allows Verifier to make Prover honestly compute $\sum_{x\in\mathcal{X}} g(x)$ for some low-degree polynomial $g(X)$ derived from the input data and some designated set $\mathcal{X}$. Verifier has no space to compute g explicitly, nor all values $\langle g(x) : x \in \mathcal{X}\rangle$, but he can afford to evaluate $g(r)$ at a *random* point r. The Prover steps in by explicitly providing $\hat{g}(X)$, a polynomial claimed to equal $g(X)$: this is cheap since g has low degree. Verifier can be convinced of this claim by checking that $\hat{g}(r) = g(r)$.

Hence, the main challenge in applying the sum-check technique is to find a way to encode the data stream problem's output as the sum of the evaluations of a low-degree polynomial g so that Verifier can, in small space, evaluate g at a random point r.

Sketches: Linearity and Beyond. A streaming Verifier evaluates $g(r)$ by suitably summarizing the input in a *sketch*. Viewing the input as updates to a data vector $\mathbf{f} = (f_1, \ldots, f_N)$, such a sketch $\mathbf{v}$ is *linear* if $\mathbf{v} = S\mathbf{f}$ for some matrix $S \in \mathbb{F}^{v\times N}$, for some field $\mathbb{F}$.[4] Typically, S is implicit in the sketching algorithm and enables stream processing in $\tilde{O}(v)$ space by translating a stream update $f_i \leftarrow f_i + \Delta$ into the sketch update $\mathbf{v} \leftarrow \mathbf{v} + \Delta S\mathbf{e}_i$, where $\mathbf{e}_i$ is the ith standard basis vector. In essentially all prior works on stream verification, the polynomial g was such that $g(r)$ could be derived from such a linear sketch $\mathbf{v}$.

There is one exception: Thaler [38] introduced an optimal $[n, n]$-scheme for Triangle-Count in which Verifier computes a *nonlinear* sketch.[5] Roughly speaking, the verifier in Thaler's protocol maintains two n-dimensional linear sketches $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$, plus a value C that is *not* a linear function of the input stream but instead depends quadratically on $\mathbf{v}^{(1)}$ and $\mathbf{v}^{(2)}$. Moreover, the jth increment to C uses information that is available while processing the jth stream update, but not after the stream is gone. This is in contrast to linear sketches themselves, where the jth sketch update depends only on the jth stream update and no others.

The Shaping Technique. Another ubiquitous idea in streaming verification is the *shaping technique*, which transforms a data vector into a multidimensional array. This trick realizes $g(X)$ as a summation of an even simpler multivariate polynomial: the latter can be evaluated directly by Verifier at several points, which forms the basis for his sketching. When applied to graph problems, this technique was historically used to reshape the $\binom{n}{2}$-dimensional vector

[4] This field is finite in the streaming verification literature, whereas traditional data streaming uses $\mathbb{R}$.
[5] Simliar nonlinearity was used recently in the more powerful model of 2-*pass schemes* [14].

of edge multiplicities. Recently, Chakrabarti and Ghosh [14] introduced the idea of reshaping the graph's *vertex space*, rather than just the edge space, thereby transforming the adjacency matrix into a 4-dimensional array. This trick was crucial to obtaining the first frugal schemes for TriangleCount and MaxMatching.

Our Contributions. The new schemes in this work make the following contributions.

- We design new polynomial encodings for the graph-theoretic problems we study.
- We prominently employ nonlinear sketches, in the above sense, for almost all of our scheme designs.
- We use the shaping technique on the vertex space, often combining it with nonlinear sketching, thus expanding the applications of this very recent innovation.

Our solutions for TriangleCount are particularly good illustrations of all of these ideas. Where Thaler's nonlinear-sketch protocol treated each vertex as monolithic, our view of each vertex as an object in $[t] \times [s]$ (for some pair t, s with $t \cdot s = n$) let us do two things. In the laconic regime, we get to use Verifier's increased space allowance in a way that Thaler's protocol cannot, thereby extending his $[n, n]$-scheme to get an optimal tradeoff. In the frugal regime, it is significantly harder to exploit vertex-space shaping because Verifier cannot even afford to devote one entry per vertex in his linear sketches. We overcome this by finding a way for many vertices to "share" each entry of each linear sketch (see the string of equations culminating in Equation (7)), thus extending Thaler's protocol to smoothly trade off communication for space.

We also extend the applicability of nonlinear sketching by identifying many further graph problems for which it yields significant improvements. Specifically, in Section 3, we describe two technical problems called InducedEdgeCountand CrossEdgeCount, which are later used as primitives to optimally solve several important graph problems, including MaxMatching. We show how to apply sum-check with a nonlinear Verifier (see, e.g, Equation (11)) to optimally solve InducedEdgeCount and CrossEdgeCount.

Finally, our schemes for SSSP feature a different kind of innovation on top of vertex-space shaping and new, clever encodings of shortest-path problems in a manner amenable to sum-check. They overcome the frugal Verifier's space limitation by exploiting the Prover's room to generate a proof stream that mimics an iterative algorithm. For the Verifier to play along with such an iterative algorithm while lacking even one bit of space per vertex, a careful layering of fingerprint-based checks is needed on top of the sum-checks. We hope that our work here opens up possibilities for other instances of porting iterative algorithms to a streaming setting with the help of a prover.

1.5 Preliminaries

In this work, the input graph, multigraph, or digraph is denoted G and defined on a fixed vertex set $V = [n]$. In the *vanilla* streaming model, G is given as a stream of (u, v) tokens, where $u, v \in V$: the token is interpreted as an insertion of edge $\{u, v\}$ or directed edge (u, v). If G is edge-weighted, the tokens are of the form (u, v, w), where $w \in \mathbb{Z}^+$ is a weight. In the *turnstile* streaming model, tokens are of the form (u, v, Δ), denoting that the quantity $\Delta \in \mathbb{Z}$ (which can be negative) is added either to the multiplicity or the weight of the edge $\{u, v\}$.

An important primitive in all our schemes is sketching a data vector by evaluating its low-degree extension at a random point. Let us explain what this means. Suppose our data vector, which has dimensionality N, is shaped into a k-dimensional array f with

dimensions $(s_1, \ldots, s_k)$, where $s_1 s_2 \cdots s_k \geqslant N$. Equivalently, we have a function f on domain $[s_1] \times \cdots \times [s_k]$. We work over a suitable finite field[6] $\mathbb{F}$. By Lagrange interpolation, there is a unique polynomial $\tilde{f}(X_1, \ldots, X_k) \in \mathbb{F}[X_1, \ldots, X_k]$ such that

- for all $(x_1, \ldots, x_k) \in [s_1] \times \cdots [s_k]$, we have $\tilde{f}(x_1, \ldots, x_k) = f(x_1, \ldots, x_k)$, and
- for all $i \in [k]$, we have $\deg_{X_i} \tilde{f} \leqslant s_i - 1$.

We call $\tilde{f}$ the low-degree $\mathbb{F}$-*extension* of f. Since $f \mapsto \tilde{f}$ is a linear map, we can write $\tilde{f}$ as a linear combination of "unit impulse" functions (also known as Lagrange basis polynomials):

$$\delta_{u_1,\ldots,u_k}(X_1, \ldots, X_k) := \prod_{i=1}^{k} \prod_{x_i \in [s_i] \setminus \{u_i\}} (u_i - x_i)^{-1}(X_i - x_i)\,. \tag{1}$$

To be precise, $\tilde{f}(X_1, \ldots, X_k) = \sum_{(u_1,\ldots,u_k) \in [s_1] \times \cdots \times [s_k]} f(u_1, \ldots, u_k)\, \delta_{u_1,\ldots,u_k}(X_1, \ldots, X_k)$. In particular, if f is built up from a stream of pointwise updates, where the jth update adds Δ_j to entry $(u_1, \ldots, u_k)_j$ of the array, then

$$\tilde{f}(X_1, \ldots, X_k) = \sum_j \Delta_j\, \delta_{(u_1,\ldots,u_k)_j}(X_1, \ldots, X_k)\,. \tag{2}$$

▶ **Fact 1.** *Given $\mathbf{p} = (p_1, \ldots, p_k) \in \mathbb{F}^k$ and a stream of pointwise updates to an initially-zero array with dimensions $(s_1, \ldots, s_k)$, we can maintain the evaluation $\tilde{f}(\mathbf{p})$ using $O(\log |\mathbb{F}|)$ space, performing $O(k)$ field arithmetic operations per update. In applications, we usually take $\mathbf{p} \in_R \mathbb{F}^k$.*[7] *For details and implementation considerations, see Cormode et al. [19].* ◀

Another useful primitive is *fingerprinting*, used prominently in our SSSP scheme and subtly in subroutines within other schemes. Its goal is to check equality between two vectors $\mathbf{a} = (a_1, \ldots, a_N)$ and $\mathbf{b} = (b_1, \ldots, b_N)$ that are provided via turnstile streams in some possibly intermixed order. This is achieved by checking that $\varphi_{\mathbf{a}}(r) = \varphi_{\mathbf{b}}(r)$ for $r \in_R \mathbb{F}$, where $\varphi_{\mathbf{a}}(X) = \sum_{j=1}^{N} a_j X^j$ is the *fingerprint polynomial* of $\mathbf{a}$ and has degree at most N. Both fingerprinting and the eventual uses of Fact 1 in sum-check protocols depend upon the following basic but powerful result.

▶ **Fact 2** (Schwartz–Zippel Lemma)**.** *For a nonzero polynomial $P(X_1, \ldots, X_n) \in \mathbb{F}[X_1, \ldots, X_n]$ of total degree d, where $\mathbb{F}$ is a finite field, $\Pr_{(r_1,\ldots,r_n) \in_R \mathbb{F}^n}[P(r_1, \ldots, r_n) = 0] \leqslant d/|\mathbb{F}|$.* ◀

At various points, we shall use a couple of schemes from Chakrabarti et al. [11, 12].

▶ **Fact 3** (SUBSET and INTERSECTION schemes; Prop. 4.1 of [12] and Thm. 5.3 of [11])**.** *Given a stream of elements of sets $S, T \subseteq [N]$ interleaved arbitrarily, for any h, v with $hv \geqslant N$, there are $[h, v]$-schemes to compute $|S \cap T|$ and to determine whether $S \subseteq T$.* ◀

2 The Triangle Counting Problem

A triangle in a (multi)graph is a set of three edges of the form $\{\{u, v\}, \{v, w\}, \{u, w\}\}$. The TRIANGLECOUNT problem asks for the number of such triangles in the input graph. We solve this problem for multigraphs given by a turnstile stream, establishing the following two theorems. The first gives improved (but possibly still not tight) tradeoffs between hcost h and vcost v in the parameter regime where $h \geqslant n$ and $v \leqslant n$. The second gives *optimal*

[6] The characteristic of $\mathbb{F}$ must be large enough to avoid "wrap around" problems under arithmetic in $\mathbb{F}$.
[7] The notation $r \in_R A$ means that r is drawn uniformly at random from the finite set A.

tradeoffs (up to logarithmic factors) in the regime where $h \leqslant n$ and $v \geqslant n$, based on the known lower bound that hv must be $\Omega(n^2)$. Both results were previously only known when $h = \Theta(n)$.

We remind the reader that parameters $t, s \in \mathbb{Z}^+$ are tunable, subject to $ts = n$.

▶ **Theorem 4** (Improved frugal schemes). *There is an $[nt^2, s]$-scheme for* TriangleCount.

▶ **Theorem 5** (Optimal tradeoff for laconic schemes). *There is a $[t, ns]$-scheme for* TriangleCount. *This is optimal up to logarithmic factors.*

Overview of Our Methods. Consider an adjacency matrix A of a graph on vertex set V. The addition of a new edge $\{u, v\}$ creates $\sum_{z \in V} A(u, z)A(v, z)$ new triangles.

Suppose that the input stream consists of L edge updates, the jth being $(v_{1j}, v_{2j}, \Delta_j)$; recall that its effect is to add Δ_j to the multiplicity of edge $\{v_{1j}, v_{2j}\}$. Suppose that the cumulative effect of the first j updates is to produce a multigraph G_j whose adjacency matrix is A_j and which has T_j triangles (counting multiplicity). As in Thaler's protocol [38], we can then account for the number of triangles added by the jth update:

$$T_j - T_{j-1} = \sum_{v_3 \in V} \Delta_j \, A_{j-1}(v_{1j}, v_3) \, A_{j-1}(v_{2j}, v_3) \, .$$

As a result, the number of triangles T in the final graph $G = G_L$ is

$$T = \sum_{j \in [L]} \sum_{v_3 \in V} \Delta_j \, A_{j-1}(v_{1j}, v_3) \, A_{j-1}(v_{2j}, v_3) \, . \tag{3}$$

Our two new families of schemes for TriangleCount apply the shaping technique to the above equation in two distinct ways, resulting in markedly different complexity behaviors.

The Laconic Schemes Regime (Proof of Theorem 5). Let $t, s \in \mathbb{N}$ be parameters with $ts = n$. We first consider rewriting the variable v_3 in Equation (3) as a pair of integers $(x_3, y_3) \in [t] \times [s]$ using some canonical bijection. This shapes each matrix A_{j-1} into a 3-dimensional array a_{j-1}, i.e., a function with domain $[n] \times [t] \times [s]$. Let $\tilde{a}$ be the $\mathbb{F}$-extension of a for a sufficiently large finite field $\mathbb{F}$ to be chosen later. Then Equation (3) becomes

$$T = \sum_{j \in [L]} \sum_{x_3 \in [t]} \sum_{y_3 \in [s]} \Delta_j \, \tilde{a}_{j-1}(v_{1j}, x_3, y_3) \, \tilde{a}_{j-1}(v_{2j}, x_3, y_3) = \sum_{x_3 \in [t]} p(x_3) \, , \quad \text{where} \tag{4}$$

$$p(X_3) = \sum_{j \in [L]} \sum_{y_3 \in [s]} \Delta_j \, \tilde{a}_{j-1}(v_{1j}, X_3, y_3) \, \tilde{a}_{j-1}(v_{2j}, X_3, y_3) \, . \tag{5}$$

By the properties of $\mathbb{F}$-extensions observed above, we have the bound $\deg p \leqslant 2(t-1)$. We now design our scheme as follows.

Stream processing. Verifier starts by picking $r_3 \in_R \mathbb{F}$. As the stream arrives, he maintains a 2-dimensional array of values $\tilde{a}_{j-1}(v, r_3, y)$, for all $(v, y) \in [n] \times [s]$, using Fact 1. He also maintains an accumulator that starts at zero and, after the jth update, is incremented by $\Delta_j \sum_{y_3 \in [s]} \tilde{a}_{j-1}(v_{1j}, r_3, y_3) \, \tilde{a}_{j-1}(v_{2j}, r_3, y_3)$. By Equation (5), the final value of this accumulator is $p(r_3)$.

Help message. Prover sends Verifier a polynomial $\hat{p}(X_3)$ of degree $\leqslant 2(t-1)$ that she claims equals $p(X_3)$.

Verification and output. Using Prover's message, Verifier computes the check value $C := \hat{p}(r_3)$ and the result value $\hat{T} := \sum_{x_3 \in [t]} \hat{p}(x_3)$. If he finds that $C \neq p(r_3)$, he outputs $\perp$. Otherwise, he believes that $\hat{p} \equiv p$ and accordingly, based on Equation (4), outputs $\hat{T}$ as the answer.

The analysis of this scheme proceeds along standard lines long established in the literature.

Error probability. An honest Prover ($\hat{p} \equiv p$) clearly ensures perfect completeness. The soundness error is the probability that Verifier's check passes despite $\hat{p} \not\equiv p$, i.e., that the random point $r_3 \in \mathbb{F}$ is a root of the nonzero degree-$(2t-2)$ polynomial $\hat{p} - p$. By the Schwartz–Zippel Lemma (Fact 2), this probability is at most $(2t-2)/|\mathbb{F}| < 1/n$, by choosing $|\mathbb{F}|$ large enough.

Help and Verification costs. Prover describes $\hat{p}$ by listing its $O(t)$ many coefficients, spending $O(t \log n)$ bits, since each is an element of $\mathbb{F}$ and $|\mathbb{F}| = n^{O(1)}$ suffices above. Verifier maintains an $n \times s$ array whose entries are in $\mathbb{F}$, for a vcost of $O(ns \log n)$. Overall, we get a $[t, ns]$-scheme, as required.

The Frugal Schemes Regime (Proof of Theorem 4). Designing frugal schemes on the basis of Equation (3) is more intricate. This time we rewrite the variables v_{1j} and v_{2j} as pairs (x_{1j}, y_{1j}) and (x_{2j}, y_{2j}), each in $[t] \times [s]$ for parameters t, s with $ts = n$. The matrices A_{j-1} are now shaped into 3-dimensional arrays b_{j-1} that can be seen as functions on the domain $[t] \times [s] \times [n]$. As before, let $\tilde{b}$ be an appropriate $\mathbb{F}$-extension. Working from Equation (3) and cleverly using the "unit impulse" function δ seen in Equation (1),

$$\begin{aligned} T &= \sum_{v_3 \in V} \sum_{j \in [L]} \Delta_j \, \tilde{b}_{j-1}(x_{1j}, y_{1j}, v_3) \, \tilde{b}_{j-1}(x_{2j}, y_{2j}, v_3) \\ &= \sum_{v_3 \in V} \sum_{w_1, w_2 \in [t]} \sum_{j \in [L]} \Delta_j \, \tilde{b}_{j-1}(w_1, y_{1j}, v_3) \, \tilde{b}_{j-1}(w_2, y_{2j}, v_3) \, \delta_{x_{1j}}(w_1) \, \delta_{x_{2j}}(w_2) \\ &= \sum_{v_3 \in V} \sum_{w_1, w_2 \in [t]} q(w_1, w_2, v_3) \,, \quad \text{where} \end{aligned} \tag{6}$$

$$q(W_1, W_2, V_3) = \sum_{j \in [L]} \Delta_j \, \tilde{b}_{j-1}(W_1, y_{1j}, V_3) \, \tilde{b}_{j-1}(W_2, y_{2j}, V_3) \, \delta_{x_{1j}}(W_1) \, \delta_{x_{2j}}(W_2) \,. \tag{7}$$

In contrast to the laconic case, we have a *multivariate* polynomial $q(W_1, W_2, V_3)$. We have the bounds $\deg_{W_1} q \leqslant 2(t-1)$, $\deg_{W_2} q \leqslant 2(t-1)$, and $\deg_{V_3} q \leqslant 2(n-1)$, for a total degree of $O(t+n) = O(n)$. Importantly, the number of monomials in q is at most $(2t-1)^2(2n-1) = O(nt^2)$. We now present the corresponding scheme and its analysis.

Stream processing. Verifier picks $r_1, r_2, r_3 \in_R \mathbb{F}$. As the stream arrives, he maintains two 1-dimensional arrays: $\tilde{b}_{j-1}(r_1, y, r_3)$ and $\tilde{b}_{j-1}(r_2, y, r_3)$, for all $y \in [s]$ (using Fact 1). He also maintains an accumulator that starts at zero and, after the jth update $(x_{1j}, y_{1j}, x_{2j}, y_{2j})$, is incremented by $\Delta_j \tilde{b}_{j-1}(r_1, y_{1j}, r_3) \tilde{b}_{j-1}(r_2, y_{2j}, r_3) \delta_{x_{1j}}(r_1) \delta_{x_{2j}}(r_2)$. By Equation (7), the final value of this accumulator is $q(r_1, r_2, r_3)$.

Notice that the accumulator is a nonlinear sketch of the input.

Help message. Prover sends Verifier a polynomial $\hat{q}(W_1, W_2, V_3)$ that she claims equals $q(W_1, W_2, V_3)$. It should satisfy the degree bounds noted above. He lacks the space to store $\hat{q}$, so she streams the coefficients of $\hat{q}$ in some canonical order.

Verification and output. As $\hat{q}$ is streamed in, Verifier computes the check value $C := \hat{q}(r_1, r_2, r_3)$ and the result value $\hat{T} := \sum_{v_3 \in [n]} \sum_{w_1, w_2 \in [t]} \hat{q}(w_1, w_2, v_3)$. If he finds that $C \neq q(r_1, r_2, r_3)$, he outputs $\perp$. Otherwise, he believes that $\hat{q} \equiv q$ and accordingly, based on Equation (6), outputs $\hat{T}$ as the answer.

Error probability. As before, we have perfect completeness and by the Schwartz–Zippel Lemma (Fact 2, this time using its full multivariate strength), this soundness error is at most $\deg q / |\mathbb{F}| = O(n)/|\mathbb{F}| < 1/n$, by choosing $|\mathbb{F}|$ large enough.

Help and Verification costs. Prover can describe $\hat{q}$ by listing its $O(nt^2)$ coefficients. Verifier maintains two s-length arrays. Overall, we get an $[nt^2, s]$-scheme, as required.

3 A Technical Result: Counting Edges in Induced Subgraphs

We introduce two somewhat technical, though still natural, graph problems: InducedEdgeCount and CrossEdgeCount. We design schemes for these problems giving optimal tradeoffs (as usual, up to logarithmic factors). These schemes are key subroutines in our schemes for more standard, well-studied graph problems – such as MaxMatching– considered in Section 4.

The InducedEdgeCount problem is defined as follows. The input is a stream of edges of a graph $G = (V, E)$ followed by a stream of vertex subsets $\langle U_1, \ldots U_\ell \rangle$ for some $\ell \in \mathbb{N}$, where $U_i \subseteq V$ for $i \in [\ell]$. To be precise, the latter portion of the stream consists of the vertices of U_1 in arbitrary order, followed by a delimiter, followed by the vertices of U_2 in arbitrary order, and so on. The desired output is $\sum_{i=1}^{\ell} |E(G[U_i])|$, the sum of the numbers of edges in the induced subgraphs $G[U_1], \ldots, G[U_\ell]$. Note that $U_1, \ldots, U_\ell$ need not be pairwise disjoint, so the sum may count some edges more than once.

The CrossEdgeCount problem is an analog of the above for induced bipartite subgraphs. The input is a stream of edges followed by ℓ *pairs* of vertex subsets $\langle (U_1, W_1), \ldots, (U_\ell, W_\ell) \rangle$, where $U_i \cap W_i = \varnothing$ for $i \in [\ell]$. The desired output is $\sum_{i=1}^{\ell} |E(G[U_i, W_i])|$, the sum of the number of cross-edges in the induced bipartite subgraphs $G[U_1, W_1], \ldots, G[U_\ell, W_\ell]$. Note that the U_is (or W_is) need not be disjoint among themselves.

Importantly, in both of these problems, the edges *precede* the vertex subsets in the stream. This makes the problems intractable in the basic data streaming model. We shall prove the following results.

▶ **Lemma 6.** *For any h, v with $hv = n^2$, there is an $[h, v]$-protocol for InducedEdgeCount.*

▶ **Lemma 7.** *For any h, v with $hv = n^2$, there is an $[h, v]$-protocol for CrossEdgeCount.*

Scheme for InducedEdgeCount (Proof of Lemma 6). For the given instance, let M denote the desired output and let A be the adjacency matrix of G. For each $i \in \ell$, let $B_i \in \{0,1\}^V$ be the indicator vector of the set U_i, i.e., $B_i(v) = 1 \iff v \in U_i$. Then,

$$M = \frac{1}{2} \sum_{i=1}^{\ell} \sum_{v_1, v_2 \in V} B_i(v_1)\, B_i(v_2)\, A(v_1, v_2)\,. \tag{8}$$

Let t, s be integer parameters such that $ts = n$. We apply the shaping technique to Equation (8) by rewriting the variables v_j as pairs of integers $(x_j, y_j) \in [t] \times [s]$, for $j \in \{1, 2\}$. This transforms the matrix A into a 4-dimensional array a and each B_i into a 2-dimensional array b_i. Let $\tilde{a}$ and $\tilde{b}_i$ be the respective $\mathbb{F}$-extensions. Equation (8) now gives

$$2M = \sum_{i=1}^{\ell} \sum_{x_1, x_2 \in [t]} \sum_{y_1, y_2 \in [s]} \tilde{b}_i(x_1, y_1)\, \tilde{b}_i(x_2, y_2)\, \tilde{a}(x_1, y_1, x_2, y_2) = \sum_{x_1, x_2 \in [t]} p(x_1, x_2)\,, \tag{9}$$

$$\text{where } \; p(X_1, X_2) = \sum_{i=1}^{\ell} \sum_{y_1, y_2 \in [s]} \tilde{b}_i(X_1, y_1)\, \tilde{b}_i(X_2, y_2)\, \tilde{a}(X_1, y_1, X_2, y_2)\,. \tag{10}$$

Our scheme exploits this expression in the same general manner as the analogous expressions for the TriangleCount schemes from Section 2 (e.g., Equation (4)). Prover sends a bivariate polynomial $\hat{p}(X_1, X_2)$, which is claimed to be p, by streaming its coefficients. Since $\deg_{X_j} p \leqslant 2(t-1)$ for $j \in \{1, 2\}$, Prover need only send $O(t^2)$ coefficients, for a help cost of $\tilde{O}(t^2)$. Verifier computes his output using Equation (9), giving perfect completeness. On

the soundness side, Verifier checks the condition $\hat{p}(r_1, r_2) = p(r_1, r_2)$ for randomly chosen $r_1, r_2 \in_R \mathbb{F}$. By the Schwartz-Zippel Lemma (Fact 2), the probability that he is fooled is at most $\deg p/|\mathbb{F}| = O(t)/|\mathbb{F}| < 1/n$, for the right choice of $\mathbb{F}$. It remains to describe how exactly Verifier evaluates $p(r_1, r_2)$, which we now address.

Processing the stream of edges. This is straightforward: Verifier maintains the 2-dimensional array of values $\tilde{a}(r_1, w, r_2, z)$, for all $w, z \in [s]$, using Fact 1.

Processing the stream of vertex subsets. Verifier initializes an accumulator to zero and allocates workspace for two arrays of length s with entries in $\mathbb{F}$. For each $i \in [\ell]$, as the vertices of U_i arrive, he maintains $\tilde{b}_i(r_1, z)$ and $\tilde{b}_i(r_2, z)$ for each $z \in [s]$, using that workspace. Upon seeing the delimiter marking the end of U_i, he computes

$$\sum_{y_1, y_2 \in [s]} \tilde{b}_i(r_1, y_1)\, \tilde{b}_i(r_2, y_2)\, \tilde{a}(r_1, y_1, r_2, y_2) \tag{11}$$

and adds this quantity to the accumulator. Note that the workspace is reused when the stream moves on from U_i to U_{i+1}. By Equation (10), after the last set U_ℓ is streamed, the accumulator holds $p(r_1, r_2)$.

Help and verification costs. We argued above that the hcost is $\tilde{O}(t^2)$. Meanwhile, Verifier's storage is dominated by the $s \times s$ array he maintains, leading to a vcost of $\tilde{O}(s^2)$.

Therefore, we obtain a $[t^2, s^2]$-scheme for any parameters t, s with $ts = n$. In other words, we get an $[h, v]$-scheme for any h, v with $hv = n^2$.

Scheme for CrossEdgeCount (Proof of Lemma 7). Our solution for InducedEdgeCount can easily be modified to obtain a protocol for CrossEdgeCount with the same costs. If B_i and C_i are the indicator vectors of the sets U_i and W_i, respectively, then the desired output is

$$M = \sum_{i=1}^{\ell} \sum_{v_1, v_2 \in V} B_i(v_1)\, C_i(v_2)\, A(v_1, v_2)\,, \tag{12}$$

where we used the fact that each $U_i \cap W_i = \varnothing$. Since Equation (12) has essentially the same form as Equation (8), a scheme very similar to the previous one solves CrossEdgeCount: Verifier simply keeps track of arrays corresponding to C_i alongside ones corresponding to B_i.

4 Maximum Matching and Other Applications of Edge Counting

In this section, we show how InducedEdgeCount and CrossEdgeCount can be used as subroutines to solve multiple problems that have been widely studied in the basic and annotated data streaming models. These problems include Maximum Matching, Triangle-Counting, Maximal Independent Set, Acyclicity Testing, Topological Sorting, and Graph Connectivity. For the frugal regime where vcost $= o(n)$, our schemes are often optimal. We specifically discuss the application to MaxMatching in Section 4.1, state the results on the other applications in Section 4.2, and give a detailed account on them in Appendix B.

4.1 The Maximum Matching Problem

We give the first optimal frugal scheme for computing the cardinality $\alpha'(G)$ of a maximum matching. As noted in prior works [14, 38], checking whether $\alpha'(G) \geqslant k$ for some k is not hard, given $\tilde{\Omega}(k)$ bits of help: Prover can simply send a matching of size k and prove its validity. The interesting part is to verify that $\alpha'(G) \leqslant k$. For this, as in prior works, we exploit the Tutte–Berge formula [9]:

$$\alpha'(G) = \frac{1}{2} \min_{U \subseteq V} \Big(|U| + |V| - \mathrm{odd}(G \setminus U) \Big), \tag{13}$$

where $\mathrm{odd}(G \setminus U)$ denotes the number of connected components in $G \setminus U$ with an odd number of vertices. Thus, to show that $\alpha'(G) \leqslant k$, Prover needs to exhibit $U^* \subseteq V$ such that $k = \frac{1}{2}(|U^*| + |V| - \mathrm{odd}(G \setminus U^*))$. Set $H := G \setminus U^*$. To verify the value of $\mathrm{odd}(H)$, the most important sub-check that Verifier must do is to check that all purported connected components of H (sent by Prover) are actually disconnected from each other. Thaler [38] gave an $[n, n]$-scheme for this subproblem (thus obtaining the first $[n, n]$-scheme for MaxMatching), while Chakrabarti and Ghosh [14] gave a $[t^3, s^2]$-scheme for any $ts = n$ (thus designing the first frugal scheme for MaxMatching, though suboptimal). The latter work notes that all other sub-checks for MaxMatching can be done by *optimal* frugal schemes (see [14], Section 4).

Optimal Frugal Scheme. To optimally check that the purported connected components of H are indeed disconnected from each other, we use the InducedEdgeCount scheme as a subroutine. Prover streams the vertices in H by listing its connected components in some order $\langle U_1, \ldots, U_\ell \rangle$. Verifier uses Lemma 6 to count $m_1 := |E(H)|$ (invoking that lemma with a single subset $V(H)$). In parallel, using the same scheme, Verifier computes the sum $m_2 = \sum_{i=1}^{\ell} |E(G[U_i])|$. The subsets U_i are pairwise disconnected iff $m_2 = m_1$, which Verifier checks. The sub-checks of whether U_is are indeed pairwise disjoint (as sets) and whether $U^* \sqcup V(H) = V(G)$ can be done via fingerprinting (as in Section 1.5).

Help and verification costs. Prover streams U^* and the vertices in H in a certain order, which adds $O(n \log n)$ bits to the hcost of the InducedEdgeCount protocol. The vcost stays the same, asymptotically, giving us an $[n + h, v]$-scheme for MaxMatching for any h, v with $hv = n^2$. Overall, we have established the following theorem.

▶ **Theorem 8.** *There is an $[nt, s]$-scheme for MaxMatching. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$ [12].*

Protocol for Space Larger Than n. There is no laconic scheme known for the general MaxMatching problem. The barrier seems to be that a natural witness for the problem is an actual maximum matching of the graph, which can be of size $\Theta(n)$. We show that large maximum matching size $\alpha'(G)$ is indeed the sole barrier to obtaining a laconic scheme. In particular, for any graph G, we give a scheme for MaxMatching with hcost $\alpha'(G)$. This yields a laconic scheme for the case when $\alpha'(G) = o(n)$. We defer the proof to Appendix A.

▶ **Theorem 9.** *For any h, v with $hv \geqslant n^2$ and $v \geqslant n$, there is an $[\alpha' + h, v]$-scheme for MaxMatching, where α' is the size of the maximum matching of the input graph. In particular, there is an $[\alpha', n^2/\alpha']$-scheme.*

4.2 Applications to Other Graph Problems

Here, we state the results we obtain for several graph problems by applying InducedEdgeCount and CrossEdgeCount. The details and proofs appear in Appendix B.

We apply the edge-counting protocols to solve the triangle-counting problem in both the standard edge arrival and the vertex arrival (adjacently list) streaming models and obtain the following theorems.

▶ **Theorem 10.** *For any h, v with $hv \geqslant n^2$, there is an $[m + h, v]$-scheme for TriangleCount. In particular, there is an $[m, n^2/m]$-scheme.*

▶ **Theorem 11.** *For any h, v with $hv \geqslant n^2$, there is an $[h, v]$-scheme for TriangleCountAdj.*

We obtain optimal frugal schemes for the fundamental graph problems of maximal independent set (MIS) and for topological sorting and acyclicity testing in directed graph streams.

▶ **Theorem 12.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for MIS. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$.*

▶ **Theorem 13.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for TopoSort. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$.*

▶ **Corollary 14.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for Acyclicity. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$.*

Finally, we show that we can apply edge-counting schemes to count the number of connected components of a graph.

▶ **Theorem 15.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for counting the number of connected components of an input graph.*

5 The Single-Source Shortest Path Problem

In the single-source shortest path (SSSP) problem, the goal is to find the distances from a source vertex v_s to every other vertex reachable from it. In Section 5.1, we give a $[Dnt, s]$-scheme for the unweighted version, whenever $ts = n$. If $s = o(n)$, Verifier does not have enough space to store the output; therefore, we aim for a protocol where Prover streams the output, and Verifier checks that it is correct using $o(n)$ space, thus achieving a frugal scheme.

In Section 5.2, we state our results for weighted SSSP for the two different weight update models (vanilla and turnstile) described in Section 1.5. The proofs follow in Appendix C.

5.1 Unweighted SSSP

We shall design a scheme that works even if the same edge appears multiple times in the stream (unlike prior work [18] that assumes that an edge appears at most once).

Prover sends distance labels $\widehat{\text{dist}}[v]$ for all $v \in V$, claiming that $\widehat{\text{dist}}[v] = \text{dist}(v_s, v)$, the actual distance from the source vertex v_s to v. Let the radius-d ball around v_s be $B_d := \{v \in V : \text{dist}(v_s, v) \leqslant d\}$ and let $\mathcal{B} := \{B_d : d \in [D]\}$ be the family of such balls. Let $\widehat{B}_d$ be the corresponding balls implied by Prover's $\widehat{\text{dist}}$ labels, and $\widehat{\mathcal{B}} := \{\widehat{B}_d : d \in [D]\}$.

To check correctness, Verifier uses fingerprinting (Section 1.5) modified as follows. Letting $B, \widehat{B}$ also denote the respective characteristic vectors, define fingerprint polynomials

$$\varphi_{\mathcal{B}}(X,Y) := \sum_{i\in[n]}\sum_{d\in[D]} B_d(i)X^iY^d\,, \quad \varphi_{\widehat{\mathcal{B}}}(X,Y) := \sum_{i\in[n]}\sum_{d\in[D]} \widehat{B}_d(i)X^iY^d\,,$$

As the $\widehat{\text{dist}}$ labels are streamed, Verifier constructs the fingerprint $\varphi_{\widehat{\mathcal{B}}}(\beta_1,\beta_2)$ for some $\beta_1,\beta_2 \in_R \mathbb{F}$.

Over the course of the protocol, using further help from Prover, Verifier will construct the sets B_d inductively and, in turn, the "actual" fingerprint $\varphi_{\mathcal{B}}(\beta_1,\beta_2)$. The next claim shows that comparing this with $\varphi_{\widehat{\mathcal{B}}}(\beta_1,\beta_2)$ validates Prover's $\widehat{\text{dist}}$ labels.

▷ Claim 16. If $\widehat{B}_d = B_d$ for all d, then $\widehat{\text{dist}}[v] = \text{dist}(v_s, v)$ for all vertices v.

Proof. Suppose not. Let d^* be the smallest d such that $\exists\, u \in B_{d^*}$ with $\widehat{\text{dist}}[u] \neq \text{dist}(v_s,u)$. Therefore, $\text{dist}(v_s,u) = d^*$. Now, d^* cannot be 0 since v_s is the only vertex in B_0 and Verifier would reject immediately if $\widehat{\text{dist}}(v_s) \neq 0$. Since $B_{d^*} = \widehat{B}_{d^*}$, we have $u \in \widehat{B}_{d^*}$. This means $\widehat{\text{dist}}(u) \leqslant d^*$. Since $\widehat{\text{dist}}(u) \neq d^*$, we have $\widehat{\text{dist}}(u) \leqslant d^* - 1$. Thus, $u \in \widehat{B}_{d^*-1}$, i.e., $u \in B_{d^*-1}$, which is a contradiction to the minimality of d^*. ◁

As before, A denotes the adjacency matrix of the graph. Putting

$$q_d(u) := \sum_{v\in V} B_d(v)\,A(v,u)\,, \text{ for each } u \in V\,, \tag{14}$$

$$\text{we have } \; B_{d+1} = \{u \in V : q_d(u) \neq 0\}\,. \tag{15}$$

To apply the shaping technique to (14), rewrite v as $(x,y) \in [t]\times[s]$. This reshapes A into a $t \times s \times n$ array $a(x,y,u)$ and B_d into a $t \times s$ array $b_d(x,y)$. As usual, let $\tilde{a}$ and $\tilde{b}_d$ be the respective $\mathbb{F}$-extensions for a suitable finite field $\mathbb{F}$. Then, Equation (14) gives

$$q_d(u) = \sum_{x\in[t]} p_d(x,u)\,, \quad \text{where} \tag{16}$$

$$p_d(X,U) := \sum_{y\in[s]} \tilde{b}_d(X,y)\,\tilde{a}(X,y,U)\,. \tag{17}$$

Stream processing. Verifier picks $r_1, r_2 \in_R \mathbb{F}$ and maintains $\tilde{a}(r_1,y,r_2)$. When he sees vertices in B_1, i.e., v_s and its neighbors, he maintains $b_1(r_1,y)$ for all $y \in [s]$ and also updates the fingerprint $\varphi_{\mathcal{B}}(\beta_1,\beta_2)$ accordingly.

Verifier wants to construct the values $b_d(r_1,y)$ inductively for $d \in [D]$. For constructing b_{d+1} values for some d, he wants all u such that $q_d(u) \neq 0$ (Equation (15)) in streaming order since he doesn't have enough space to either store the entire polynomial of degree $n-1$ that agrees with q_d (so as to go over all evaluations), or to parallelly evaluate it at n values while its coefficients are streamed. Hence, he asks for the following help message.

Help message processing. Prover continues her proof stream by sending $\langle \hat{p}_1, Q_1, \ldots, \hat{p}_D, Q_D\rangle$, where $Q_d := \langle \hat{q}_d(u) : u \in V\rangle$, claiming that $\hat{p}_d \equiv p_d$ and $\hat{q}_d(u) = q_d(u)$ for each $d \in [D]$ and $u \in [n]$.

While $\hat{p}_d$ is streamed, Verifier computes the following in parallel:

- $\hat{p}_d(r_1,r_2)$;
- $p_d(r_1,r_2)$, using Equation (17);
- the fingerprint $g_d := \sum_{u\in[n]}\sum_{x\in[t]} \hat{p}_d(x,u)\beta^u$ (for some $\beta \in_R \mathbb{F}$).

After reading $\hat{p}_d$, he checks whether $\hat{p}_d(r_1, r_2) = p_d(r_1, r_2)$. If so, he believes that $\hat{p}_d \equiv p_d$ and, in turn, that $g_d = \sum_{u\in[n]} q_d(u)\beta^u$ (by Equation (16)). Next, as Q_d is streamed,

- Verifier computes the fingerprint $g'_d := \sum_{u\in[n]} \hat{q}_d(u)\beta^u$.
- For each u with $\hat{q}_d(u) \neq 0$, due to Equation (15) (and assuming for now that the $\hat{q}_d$ values are correct), he treats u as a stream update for B_{d+1}, and (i) maintains $b_{d+1}(r_1, y)$ for all $y \in [s]$, and (ii) accordingly updates the fingerprint $\varphi_{\mathcal{B}}(\beta_1, \beta_2)$.

After reading Q_d, he checks if the fingerprints g_d and g'_d match. If they do, he believes that all $\hat{q}_d$ values in Q_d were correct and hence, the b_{d+1} values he constructed are correct as well. He moves on to the next iteration, i.e., starts reading $\hat{p}_{d+1}$.

Final Verification. After the Dth iteration, Verifier checks if the two fingerprints $\varphi_{\mathcal{B}}(\beta_1, \beta_2)$ and $\varphi_{\widehat{\mathcal{B}}}(\beta_1, \beta_2)$ match. If the check passes, then he believes that the $\widehat{\text{dist}}$ labels were correct, at least upto distance D (by Claim 16). Finally, he checks if fingerprints for B_D and B_{D+1} match to verify that vertices in $V \setminus B_D$ are indeed unreachable.

Error probability. Verifier does $O(D)$ fingerprint-checks and $O(D)$ sum-checks, using degree-$O(n)$ polynomials. Using $|\mathbb{F}| > n^3$ (and a union bound), the soundness error is $< 1/n$.

Help and verification costs. The set of $\widehat{\text{dist}}$ labels sent by the Prover has size $\tilde{O}(n)$. Each polynomial $\hat{p}_d$ has nt monomials and each Q_d has $O(n)$ field elements, and hence, size $\tilde{O}(n)$. Therefore, the total hcost is $\tilde{O}(Dnt)$. Initially, the $\tilde{A}$ and $\tilde{b}_1$ values are stored using $\tilde{O}(s)$ space. Next, the $\tilde{b}_d$ and g_d values are maintained reusing space of b_{d-1} and g_{d-1} values respectively. We also use $O(1)$ many other fingerprints that take $O(\log n)$ space each. Hence, the total vcost is $\tilde{O}(s)$.

▶ **Theorem 17.** *There is a $[Dnt, s]$-scheme for unweighted SSSP, where $D = \max_{v\in V} \text{dist}(v_s, v)$.*

▶ **Corollary 18.** *There is a $[Knt, s]$-scheme for st-ShortestPath, where $K = \text{dist}(v_s, v_t)$.*

Proof. The protocol for SSSP incurs a factor of D in the hcost since it constructs B_d for each $d \in [D]$. For the simpler st-ShortestPath problem, we can inductively construct balls and stop as soon as we find the destination vertex v_t in some B_d (i.e., get $\hat{q}_{d-1}(v_t) \neq 0$). We must find it in B_K where K is the length of a shortest v_s–v_t path. Thus, we will only incur a factor of K in the hcost, which implies a $[Knt, s]$-scheme for st-ShortestPath. ◀

Thus, we generalize the $[Dnt, s]$-scheme of Cormode et al. [18] from st-ShortestPath to SSSP. Our result for st-ShortestPath generalizes the $[Kn, n]$-scheme of Chakrabarti and Ghosh [14] by giving a smooth tradeoff and also improves upon the $[Dnt, s]$-scheme of Cormode et al. [18], since K can be arbitrarily smaller than D.

5.2 Weighted SSSP

We consider the more general weighted version of SSSP in the turnstile and the vanilla weight update models (see Section 1.5 for the definitions) and obtain the following results. The details of the schemes along with the proofs of correctness appear in Appendix C.

▶ **Theorem 19.** *There is a $[DWn, n]$-scheme for weighted SSSP in the turnstile weight update model.*

▶ **Theorem 20.** *There is a $[Dn, Wn]$-scheme for weighted SSSP in the vanilla streaming model.*

References

1 Amirali Abdullah, Samira Daruki, Chitradeep Dutta Roy, and Suresh Venkatasubramanian. Streaming verification of graph properties. In *Proc. 27th International Symposium on Algorithms and Computation*, pages 3:1–3:14, 2016.

2 Kook Jin Ahn, Sudipto Guha, and Andrew McGregor. Analyzing graph structure via linear measurements. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 459–467, 2012.

3 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, 1998. Preliminary version in *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 14–23, 1992.

4 Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *J. ACM*, 45(1):70–122, 1998. Preliminary version in *Proc. 33rd Annual IEEE Symposium on Foundations of Computer Science*, pages 2–13, 1992.

5 Sepehr Assadi, Yu Chen, and Sanjeev Khanna. Sublinear algorithms for $(\Delta+1)$ vertex coloring. In *Proc. 30th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 767–786, 2019.

6 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proc. 28th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742, 2017.

7 Ziv Bar-Yossef, Ravi Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proc. 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 623–632, 2002.

8 Suman K. Bera and Amit Chakrabarti. Towards Tighter Space Bounds for Counting Triangles and Other Substructures in Graph Streams. In *34th Symposium on Theoretical Aspects of Computer Science (STACS 2017)*, pages 11:1–11:14, 2017.

9 J.A. Bondy and U.S.R Murty. *Graph Theory*. Springer Publishing Company, Incorporated, 1st edition, 2008.

10 Luciana S. Buriol, Gereon Frahling, Stefano Leonardi, Alberto Marchetti-Spaccamela, and Christian Sohler. Counting triangles in data streams. In *Proc. 25th ACM Symposium on Principles of Database Systems*, pages 253–262, 2006.

11 Amit Chakrabarti, Graham Cormode, Navin Goyal, and Justin Thaler. Annotations for sparse data streams. In *Proc. 25th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 687–706, 2014.

12 Amit Chakrabarti, Graham Cormode, Andrew McGregor, and Justin Thaler. Annotations in data streams. *ACM Trans. Alg.*, 11(1):Article 7, 2014.

13 Amit Chakrabarti, Graham Cormode, Andrew McGregor, Justin Thaler, and Suresh Venkatasubramanian. Verifiable stream computation and Arthur-Merlin communication. In *Proc. 30th Annual IEEE Conference on Computational Complexity*, pages 217–243, 2015.

14 Amit Chakrabarti and Prantar Ghosh. Streaming verification of graph computations via graph structure. In *Proc. 33rd International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 70:1–70:20, 2019.

15 Amit Chakrabarti, Prantar Ghosh, Andrew McGregor, and Sofya Vorotnikova. Vertex ordering problems in directed graph streams. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020*, pages 1786–1802, 2020.

16 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Math. Program.*, 154(1–2):225–247, 2015. Preliminary version in *Proc. 17th Conference on Integer Programming and Combinatorial Optimization*, pages 210–221, 2014.

17 Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. In *Proc. 46th International Colloquium on Automata, Languages and Programming*, pages 45:1–45:14, 2019.

18 Graham Cormode, Michael Mitzenmacher, and Justin Thaler. Streaming graph computations with a helpful advisor. *Algorithmica*, 65(2):409–442, 2013.

19 Graham Cormode, Justin Thaler, and Ke Yi. Verifying computations with streaming interactive proofs. *Proc. VLDB Endowment*, 5(1):25–36, 2011.

20 Alireza Farhadi, Mohammad Taghi Hajiaghayi, Tung Mai, Anup Rao, and Ryan A. Rossi. Approximate maximum matching in random streams. In *Proc. 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1773–1785, 2020.

21 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(6):1709–1727, 2008. Preliminary version in *Proc. 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 745–754, 2005.

22 Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proc. 23rd Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 468–485, 2012.

23 Tom Gur and Ran Raz. Arthur–Merlin streaming complexity. In *Proc. 40th International Colloquium on Automata, Languages and Programming*, pages 528–539, 2013.

24 Madhav Jha, C. Seshadhri, and Ali Pinar. A space efficient streaming algorithm for triangle counting using the birthday paradox. In *Proc. 19th Annual SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 589–597, 2013.

25 Hossein Jowhari and Mohammad Ghodsi. New streaming algorithms for counting triangles in graphs. In *Computing and Combinatorics*, pages 710–716. Springer Berlin Heidelberg, 2005.

26 John Kallaugher, Andrew McGregor, Eric Price, and Sofya Vorotnikova. The complexity of counting cycles in the adjacency list streaming model. In *Proc. 38th ACM Symposium on Principles of Database Systems*, pages 119–133, 2019.

27 Daniel M. Kane, Kurt Mehlhorn, Thomas Sauerwald, and He Sun. Counting arbitrary subgraphs in data streams. In *Automata, Languages, and Programming*, pages 598–609. Springer Berlin Heidelberg, 2012.

28 Michael Kapralov. Better bounds for matchings in the streaming model. In *Proc. 24th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1679–1697, 2013.

29 Michael Kapralov, Slobodan Mitrovic, Ashkan Norouzi-Fard, and Jakab Tardos. Space efficient approximation to maximum matching size from uniform edge samples. In *Proc. 31st Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1753–1772, 2020.

30 Hartmut Klauck and Ved Prakash. Streaming computations with a loquacious prover. In *Proc. 4th Conference on Innovations in Theoretical Computer Science*, pages 305–320, 2013.

31 Mihail N. Kolountzakis, Gary L. Miller, Richard Peng, and Charalampos E. Tsourakakis. Efficient triangle counting in large graphs via degree-based vertex partitioning. *Internet Mathematics*, 8(1-2):161–185, 2012.

32 Carsten Lund, Lance Fortnow, Howard J. Karloff, and Noam Nisan. Algebraic methods for interactive proof systems. *J. ACM*, 39(4):859–868, 1992.

33 Andrew McGregor. Finding graph matchings in data streams. In *Proc. 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, pages 170–181, 2005.

34 Andrew McGregor, Sofya Vorotnikova, and Hoa T. Vu. Better algorithms for counting triangles in data streams. In *Proc. 35th ACM Symposium on Principles of Database Systems*, pages 401–411, 2016.

35 Michael Mitzenmacher and Justin Thaler. Technical perspective: Catching lies (and mistakes) in offloaded computation. *Commun. ACM*, 59(2):102, 2016.

36 Adi Shamir. IP = PSPACE. *J. ACM*, 39(4):869–877, 1992.

37 Justin Thaler. Data stream verification. In *Encyclopedia of Algorithms*, pages 494–499. Springer Berlin Heidelberg, 2016.

38 Justin Thaler. Semi-streaming algorithms for annotated graph streams. In *Proc. 43rd International Colloquium on Automata, Languages and Programming*, pages 59:1–59:14, 2016.

39 Peter A. Tucker, David Maier, Lois M. L. Delcambre, Tim Sheard, Jennifer Widom, and Mark P. Jones. Punctuated data streams, 2005.

40 Douglas B. West. *Introduction to Graph Theory*. Prentice Hall, 2nd edition, 2001.

41 Ke Yi, Feifei Li, Marios Hadjieleftheriou, George Kollios, and Divesh Srivastava. Randomized synopses for query assurance on data streams. In *Proc. 24th International Conference on Data Engineering*, pages 416–425, 2008.

A Maximum Matching Redux (Proof of Theorem 9)

In Section 4.1, we gave a frugal scheme for MaxMatching, i.e., one that has hcost $\geqslant n$ and vcost $\leqslant n$. Here, we show that we can get a laconic scheme, i.e., one with hcost $= o(n)$ and vcost $= \omega(n)$ as long as the maximum matching size is $o(n)$.

Let $H = G \setminus U^*$, where U^* is as in the MaxMatching protocol in Section 4.1, and let $U_1, \ldots, U_\ell$ be the connected components of H. By the Tutte-Berge formula (Equation (13)), we have $2k = |U^*| + (n - \text{odd}(H))$. This leads to the following observations.

▶ **Observation 21.** $|U^*| = O(k)$.

▶ **Observation 22.** *The number of edges in a spanning forest of H is $|V(H)| - \ell \leqslant n - \text{odd}(H) = O(k)$.*

We now describe our protocol, which is along the lines of the protocol in Section 4.1, but this time we crucially use the fact that we are allowing Verifier a space usage of $v \geqslant n$.

To show that $\alpha'(G) \geqslant k$, Prover sends a matching M of size k. Verifier stores M explicitly and checks that it is indeed a matching. Then, he verifies that $M \subseteq E$ using the Subset Scheme (Fact 3). Therefore, this part of the scheme uses hcost $\tilde{O}(k+h)$ and vcost $\tilde{O}(v)$ for any h, v with $hv = n^2$ and $v \geqslant n$.

Recall from Section 4.1 that to show that $\alpha'(G) \leqslant k$, it suffices to compute $\text{odd}(H)$. Prover sends the set U^*. By Observation 21, this takes $\tilde{O}(k)$ hcost. Verifier has $\Omega(n)$ space, and hence, he can store $V \setminus U^* = V(H)$. Next, Prover sends a spanning forest F of H. By Observation 22, this again incurs hcost $\tilde{O}(k)$. Verifier stores F and verifies that $F \subseteq E$ using the Subset Scheme (Fact 3). From F, Verifier explicitly knows the purported connected components $U_1, \ldots, U_\ell$ of H. He finally verifies that U_i's are disconnected from each other by checking that all edges in H are contained in these components. He can do this by checking whether $|E \cap (V(H) \times V(H))| = |E \cap (\cup_{i=1}^{\ell} U_i \times U_i)|$ using the Intersection Scheme (Fact 3). If the check passes he goes over the U_is to compute $\text{odd}(H)$ and thus, this part can also be solved using a $[k+h, v]$ scheme for any h, v with $hv = n^2$ and $v \geqslant n$. Hence, we obtain the following theorem.

▶ **Theorem 9.** *For any h, v with $hv \geqslant n^2$ and $v \geqslant n$, there is an $[\alpha' + h, v]$-scheme for MaxMatching, where α' is the size of the maximum matching of the input graph. In particular, there is an $[\alpha', n^2/\alpha']$-scheme.*

B Applications of Edge-Counting to Other Graph Problems

In Section 4.1, we used a scheme for InducedEdgeCount to obtain an optimal frugal scheme for MaxMatching. Below, we give applications of edge-counting schemes to several other well-studied graph problems.

Triangle-Counting. A scheme for TriangleCount follows immediately from InducedEdgeCount. For $v \in [n]$, set the subsets $U_v = N(v)$, the neighborhood of vertex v. Then, observe that InducedEdgeCount returns three times the total number of triangles in the graph. The sets U_v, however, need to be sent in some order by Prover, and so the additional hcost to InducedEdgeCount is $\tilde{O}\left(\sum_v |N(v)|\right) = \tilde{O}(m)$. As Prover basically repeats the edge stream in a different order, we can check if it's consistent with the input stream by fingerprinting (see Section 1.5). Hence, we get an $[m + h, v]$-scheme for any h, v with $hv = n^2$.

▶ **Theorem 10.** *For any h, v with $hv \geqslant n^2$, there is an $[m + h, v]$-scheme for TriangleCount. In particular, there is an $[m, n^2/m]$-scheme.*

The only other scheme for TriangleCount achieving $hv = n^2$ tradeoff with vcost $= o(n)$ was an $[n^2, 1]$-scheme by Chakrabarti et al. [12]. Our result generalizes it for any graph with m edges, thus achieving a better hcost and a smooth tradeoff for sparse graphs.

We note that in the above scheme, Prover needs to send the sets $U_v = N(v)$ because the InducedEdgeCount protocol needs the neighborhood of each vertex to arrive contiguously in the stream. This is essentially the input stream order in the *adjacency-list* or the *vertex-arrival* streaming model. Thus, for the problem TriangleCount-Adj, Verifier gets the U_vs in the desired order as part of the input; so Prover need not repeat them, saving the huge $\tilde{O}(m)$ hcost. However, there is another issue in directly applying the InducedEdgeCount subroutine in this case. In the definition of InducedEdgeCount, we assume that all the edges in the graph arrive before the vertex subsets U_i. Here, the U_vs and the edges arrive in interleaved manner (although each U_v arrives contiguously). But we show that we can still apply the scheme for InducedEdgeCount to get the desired output. Let the order in which the U_vs appear be $\langle U_1, \ldots U_n \rangle$, and let G_v denote the graph consisting of edges seen till the arrival of $U_v = N(v)$. Then, applying InducedEdgeCount, what we count is

$$\sum_{v \in [n]} |E(G_v[N(v)])| = \sum_{v \in [n]} \#\{\text{triangles incident on } v \text{ in } G_v\} = 2T\,.$$

The last equality follows since every triangle whose vertices appear in the order $\langle v_1, v_2, v_3 \rangle$ will be counted twice: once when v_2 arrives and once when v_3 arrives. We therefore obtain the following theorem.

▶ **Theorem 11.** *For any h, v with $hv \geqslant n^2$, there is an $[h, v]$-scheme for TriangleCount-Adj.*

Maximal Independent Set (MIS). Recent works [5, 17] have studied the problem of finding a maximal independent set in the basic data streaming model. They show a lower bound of $\Omega(n^2)$ for a one-pass streaming algorithm. This implies a lower bound of $hv \geqslant n^2$ for any $[h, v]$-scheme for MIS. Hence, we aim for $hv = n^2$ and describe a frugal scheme using InducedEdgeCount. Since the output size of the problem can be $\Theta(n)$, it would only make sense in the frugal regime if the Prover sends the output as a stream and the Verifier checks that it is valid using $o(n)$ space.

Let U be an MIS in the graph G. Prover sends U and Verifier uses InducedEdgeCount to count the number of edges in $G[U]$ and verifies that it equals 0. If the check passes, U is indeed an independent set. It remains to check the maximality of U. If U is maximal, then, for each vertex v in $G \setminus U$, there must be a vertex u in U, such that (v, u) is an edge. Prover points out such a vertex $u \in U$ for each $v \in G \setminus U$. Let F denote this set of $|G \setminus U|$

purported edges. Now, we use Subset Scheme (Fact 3) to verify that $F \subseteq E$, i.e., all these edges are actually present in G. We can use fingerprinting (as in Section 1.5) to check that F contains an edge for each vertex in $G \setminus U$ and the Intersection Scheme to verify that the set of their partners is disjoint from $G \setminus U$, i.e., belong to U. Thus, the additional hcost to InducedEdgeCount, Subset, and Intersection Schemes is $\tilde{O}(n)$, the number of bits required to send U and F. Therefore, by Lemma 6, we get an $[n+h, v]$-scheme for MIS for any h, v with $hv = n^2$. Thus, our scheme is optimal for the frugal regime.

▶ **Theorem 12.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for MIS. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$.*

Acyclicity Testing and Topological Sorting. We now turn to the Acyclicity problem in directed graphs. It is easy to prove that a graph is *not* acyclic by showing the existence of a cycle C. Verifier checks that $C \subseteq E$ using Subset Scheme (Fact 3). Hence, this can be done using an $[h, v]$-scheme for any $h \geqslant |C|$.

The more interesting case is when the graph is indeed acyclic. Note that a directed graph is acyclic if and only if it has a topological ordering. Thus, it suffices to show a valid topological ordering of the vertices. TopoSort is a fundamental graph algorithmic problem of independent interest. Acyclicity has a one-pass lower bound of $\Omega(n^2)$ in the basic data streaming model. Recently, Chakrabarti et al. [15] showed that TopoSort also requires $\Omega(n^2)$ space in one pass. These translate to a lower bound of $hv \geqslant n^2$ for any $[h, v]$-scheme for these problems. Hence, we aim for a scheme with $hv = n^2$ and design a protocol for TopoSort in the frugal regime. Since this problem has output size $\tilde{\Theta}(n)$, we aim for a protocol where Prover sends a topological ordering of the graph and Verifier checks its validity using $o(n)$ space. Moreover, this protocol can be used for the YES case of Acyclicity.

Verifier uses CrossEdgeCount to solve this. As Prover sends the topological order $\langle v_1, \ldots, v_n \rangle$, for each $i \in [n-1]$, Verifier sets $U_i = \{v_1, \ldots, v_i\}$ and $W_i = \{v_{i+1}\}$ for CrossEdgeCount. Thus, the protocol counts precisely the number of forward edges induced by the ordering. If it equals m, then the ordering is indeed a valid topological order. Note that since $U_{i+1} = U_i \cup \{v_{i+1}\}$, Prover doesn't need to send U_{i+1} afresh; just v_{i+1} is enough for Verifier to update his sketch. Verifier can use fingerprinting (see Section 1.5) to make sure that precisely the set V was sent in some order. Hence, the additional hcost to CrossEdgeCount is the number of bits required to express the topological order, i.e., $\tilde{O}(n)$. Therefore, by Lemma 7, we get a $[n+h, v]$-scheme for any $hv = n^2$.

▶ **Theorem 13.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for TopoSort. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$.*

▶ **Corollary 14.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for Acyclicity. This is optimal up to logarithmic factors, since any (h, v)-scheme is known to require $hv = \Omega(n^2)$.*

For dense graphs, our result generalizes the $[m, 1]$-scheme of Cormode et al. [18] for Acyclicity by achieving a smooth tradeoff.

Graph Connectivity. The graph connectivity problem has garnered considerable attention in the basic and annotated streaming settings [2, 12, 38]. For any t, s with $ts = n$, Chakrabarti et al. [12] gave an $[nt, s]$-scheme that determines whether an input graph is connected or not. Their scheme cannot, however, solve the more general problem of returning the number of connected components. The $[t^3, s^2]$-scheme (for any $ts = n$) of Chakrabarti and Ghosh [14] does solve this problem, but has a worse tradeoff. As noted in Section 4.1, we can

use InducedEdgeCount to check that all purported connected components are indeed disconnected from each other. On the other hand, the scheme of Chakrabarti et al. [12] can check whether each component is actually connected. Hence, we can verify the number of connected components claimed by Prover by running these schemes parallelly. Thus, we generalize the result of Chakrabarti et al. [12] by obtaining an $[nt, s]$-scheme for counting the number of connected components of a graph.

▶ **Theorem 15.** *For any t, s with $ts = n$, there is an $[nt, s]$-scheme for counting the number of connected components of an input graph.*

C Details of Weighted SSSP Schemes

Here, we give full details of the schemes for weighted SSSP in our two streaming models, thereby proving Theorems 19 and 20.

Turnstile weight update (Proof of Theorem 19). Assume that the edge weights are positive integers. Each stream update increments/decrements the weight of an edge. The *distance* from vertex u to vertex v refers to the weight of the shortest path from u to v. Let D be the longest distance from the source s to any other vertex reachable from it, and W be the maximum weight of an edge.

Define

$$\delta_w(X) := \prod_{\substack{w' \in [W] \\ w' \neq w}} (X - w') \Big/ \prod_{\substack{w' \in [W] \\ w' \neq w}} (w - w') .$$

Let A denote the adjacency matrix of the weighted graph G, i.e., $A(u, v)$ is the weight of the edge (u, v). Let B_d (resp. N_d) denote the set of vertices at a distance of at most (resp. exactly) d from the source vertex v_s. Then,

$$N_{d+1} = \{u \in V \setminus B_d : p_d(u) \neq 0\} , \tag{18}$$

$$\text{where } \; p_d(U) = \sum_{v \in B_d} \delta_{w(v)}(\tilde{A}(v, U)) \;\text{ and }\; w(v) = d + 1 - \text{dist}[v] . \tag{19}$$

Stream processing. Verifier chooses $r \in_R \mathbb{F}$ and maintains $\tilde{A}(v, r)$ for all v. He stores B_1 with dist$[v]$ labelled as 1 for each $v \in B_1$.

Help message processing and verification. Prover sends polynomials $\hat{p}_d$ and claims that $\hat{p}_d \equiv p_d$ for each $d \in [D]$. Verifier computes B_d inductively for $d \in [D]$ as follows.

Assume that, for some $d \in [D - 1]$, he has the set B_d with dist$[v]$ labeled on each vertex $v \in B_d$; this holds initially as he has stored B_1. He computes $p_d(r)$ using Equation (19) and checks whether $\hat{p}_d(r) = p_d(r)$. If the check passes, he believes that $\hat{p}_d \equiv p_d$ and evaluates $\hat{p}_d(u)$ for each $u \in V \setminus B_d$ and constructs N_{d+1} using Equation (18). Then, B_{d+1} is given by $N_{d+1} \uplus B_d$.

After B_D is obtained, we get all vertices reachable from s along with their distances from s. Finally, Verifier checks if the other vertices are indeed unreachable from s by verifying that there is no cross-edge between B_D and $V \setminus B_D$, i.e., if $E \cap (B_D \times (V \setminus B_D)) = \varnothing$. (Intersection scheme, see Fact 3)

Error probability. Verifier uses the same element r for $O(D)$ invocations of the sum-check protocol, where each application of the sum-check protocol is to a univariate polynomial of degree $O(Wn)$. Choosing $|\mathbb{F}| > DWn^2$, the soundness error for each invocation of the sum-check protocol is at most $1/(Dn)$. Taking a union bound over all $O(D)$ invocations, we get that the total error probability of the protocol is at most $O(1/n)$.

Help and verification costs We have $\deg p_d = O(Wn)$ for each $d \in [D]$ and hence, hcost is $\tilde{O}(DWn)$. Verifier needs to store all vertices and $\tilde{A}(v, r)$ for each $v \in [n]$, and hence, vcost is $\tilde{O}(n)$. The final disjointness can be checked by an $[n, n]$ intersection scheme.

▶ **Theorem 19.** *There is a $[DWn, n]$-scheme for weighted SSSP in the turnstile weight update model.*

Vanilla Stream (Proof of Theorem 20). We now describe a protocol for SSSP in the model where the edges arrive with their weights, without any further update on them. This is the "vanilla" streaming model.

At the end of the stream, Prover sends the distances dist$[v]$ and prev$[v]$ – the parent of v in the shortest path tree rooted at s – for all $v \in V$. Verifier checks whether the edges and their weights implied by this proof are correct, using a $[Wn, n]$ subset scheme. Thus, if Prover is honest, we get the distance as well as shortest path from s to each vertex. But we also need to check that there is no path to any vertex shorter than the ones claimed by Prover. We describe a protocol for this.

For $u, v \in V$ and $w \in [W]$, define the indicator function f as $f(u, v, w) = 1$ iff $A(u, v) = w$. Let $\tilde{f}$ be the $\mathbb{F}$-extension of f, for some large finite field $\mathbb{F}$.

Retain the definitions of B_d and N_d from last section with the definition of the polynomial p_d changed to

$$p_d(U) = \sum_{v \in B_d} \tilde{f}(v, U, d + 1 - \text{dist}_s[v]) \tag{20}$$

Hence, it still holds that

$$N_{d+1} = \{u \in V \setminus B_d : p_d(u) \neq 0\} \ . \tag{21}$$

Stream processing. The stream updates are of the form (u, v, w) denoting that $A(u, v) = w$. Verifier picks $r \in_R \mathbb{F}$ and maintains $\tilde{f}(v, r, w)$ for each $v \in V$ and $w \in [W]$. He also stores the set B_1 with dist$_s$ labels set to 1 for each vertex in the set.

Help message processing and verification. This part is similar to the turnstile weight update protocol. Of course, this time, the Verifier computes $p_d(r)$ using Equation (20).

Error probability. Each polynomial p_d has degree $O(n)$. Verifier does sum-checks for $O(D)$ such polynomials. Choosing $|\mathbb{F}| \gg Dn$, we can make the error probability small by union bound.

Help and Verification costs. Since the degree of each p_d is at most n, the total hcost is $\tilde{O}(Dn)$. Verifier stores $\tilde{f}(v, r, w)$ for each $v \in V$ and $w \in [W]$, which requires $\tilde{O}(Wn)$ space. We also need to store all vertices as we go on assigning the distance labels. Hence, the total vcost of this protocol is $\tilde{O}(Wn)$.

▶ **Theorem 20.** *There is a $[Dn, Wn]$-scheme for weighted SSSP in the vanilla streaming model.*

Disjointness Through the Lens of Vapnik–Chervonenkis Dimension: Sparsity and Beyond

Anup Bhattacharya
Indian Statistical Institute, Kolkata, India
bhattacharya.anup@gmail.com

Sourav Chakraborty
Indian Statistical Institute, Kolkata, India
sourav@isical.ac.in

Arijit Ghosh
Indian Statistical Institute, Kolkata, India
arijitiitkgpster@gmail.com

Gopinath Mishra
Indian Statistical Institute, Kolkata, India
gopianjan117@gmail.co

Manaswi Paraashar
Indian Statistical Institute, Kolkata, India
manaswi.isi@gmail.co

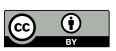
Abstract

The disjointness problem – where Alice and Bob are given two subsets of $\{1, \ldots, n\}$ and they have to check if their sets intersect – is a central problem in the world of communication complexity. While both deterministic and randomized communication complexities for this problem are known to be $\Theta(n)$, it is also known that if the sets are assumed to be drawn from some restricted set systems then the communication complexity can be much lower. In this work, we explore how communication complexity measures change with respect to the complexity of the underlying set system. The complexity measure for the set system that we use in this work is the Vapnik–Chervonenkis (VC) dimension. More precisely, on any set system with VC dimension bounded by d, we analyze how large can the deterministic and randomized communication complexities be, as a function of d and n. The d-sparse set disjointness problem, where the sets have size at most d, is one such set system with VC dimension d. The deterministic and the randomized communication complexities of the d-sparse set disjointness problem have been well studied and is known to be $\Theta\left(d \log\left(n/d\right)\right)$ and $\Theta(d)$, respectively, in the multi-round communication setting. In this paper, we address the question of whether the randomized communication complexity is always upper bounded by a function of the VC dimension of the set system, and does there always exist a gap between the deterministic and randomized communication complexity for set systems with small VC dimension.

In this paper, we construct two natural set systems of VC dimension d, motivated from geometry. Using these set systems we show that the deterministic and randomized communication complexity can be $\widetilde{\Theta}\left(d \log\left(n/d\right)\right)$ for set systems of VC dimension d and this matches the deterministic upper bound for all set systems of VC dimension d. We also study the deterministic and randomized communication complexities of the set intersection problem when sets belong to a set system of bounded VC dimension. We show that there exists set systems of VC dimension d such that both deterministic and randomized (one-way and multi-round) complexities for the set intersection problem can be as high as $\Theta\left(d \log\left(n/d\right)\right)$, and this is tight among all set systems of VC dimension d.

2012 ACM Subject Classification Theory of computation → Communication complexity

Keywords and phrases Communication complexity, VC dimension, Sparsity, and Geometric Set System



© Anup Bhattacharya, Sourav Chakraborty, Arijit Ghosh, Gopinath Mishra, and Manaswi Paraashar; licensed under Creative Commons License CC-BY

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 23; pp. 23:1–23:15



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

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.23

Category RANDOM

Funding *Anup Bhattacharya*: Supported by SERB-National Post Doctoral Fellowship, India.
Arijit Ghosh: Supported by Ramanujan Fellowship (No. SB/S2/RJN-064/2015), India.

Acknowledgements The authors would like to thank Sudeshna Kolay and Arijit Bishnu for the many discussions in the early stages of this work.

1 Introduction

Since its introduction by Yao [22], communication complexity occupies a central position in theoretical computer science. A striking feature of communication complexity is its interplay with other diverse areas like analysis, combinatorics, and geometry [9, 17]. Vapnik and Chervonenkis [21] introduced the measure *Vapnik-Chervonenkis dimension* or *VC dimension* for set systems in the context of statistical learning theory. As was the case with communication complexity, VC dimension has found numerous connections and applications in many different areas like approximation algorithms, discrete and combinatorial geometry, computational geometry, discrepancy theory and many other areas [10, 3, 14, 11]. In this work we study both of them under the same lens: of restricted systems and, for the first time, prove that geometric simplicity does not necessarily imply efficient communication complexity.

Lets start with recollecting some definitions from Vapnik–Chervonenkis theory. Let $\mathcal{S}$ be a collection of subsets of a *universe* $\mathcal{U}$. For a subset y of $\mathcal{U}$, we define

$$\mathcal{S}|_y := \{y \cap x \,:\, x \in \mathcal{S}\}\,.$$

We say a subset y of $\mathcal{U}$ is *shattered* by $\mathcal{S}$ if $\mathcal{S}|_y = 2^y$, where 2^y denotes the power set of y. *Vapnik–Chervonenkis (VC) dimension* of $\mathcal{S}$, denoted as VC-dim($\mathcal{S}$), is the size of the largest subset y of $\mathcal{U}$ shattered by $\mathcal{S}$. VC dimension has been one of the fundamental measures for quantifying complexity of a collection of subsets.

Now let us revisit the world of communication complexity. Let $f : \Omega_1 \times \Omega_2 \to \Omega$. In *communication complexity*, two players Alice and Bob get as inputs $x \in \Omega_1$ and $y \in \Omega_2$ respectively, and the goal for the players is to device a protocol to compute $f(x, y)$ by exchanging as few bits of information between themselves as possible.

The *deterministic communication complexity* $D(f)$ of a function f is the minimum number of bits Alice and Bob will exchange in the worst case to deterministically compute the function f. In the randomized setting, both Alice and Bob share an infinite random source[1] and the goal is to give the correct answer with probability at least 2/3. The randomized communication complexity $R(f)$ of f denotes the minimum number of bits exchanged by the players in the worst case input by the best randomized protocol computing f. In both deterministic and randomized settings, Alice and Bob are allowed to make multiple rounds of interaction. Communication complexity when the number of rounds of interaction is bounded is also often studied. An important special case is when only one round of communication is allowed, that is, only Alice is allowed to send messages to Bob and Bob computes the output. We will denote by $D^{\rightarrow}(f)$ and $R^{\rightarrow}(f)$ the *one way deterministic communication complexity* and *one way randomized communication complexity* respectively, of f.

[1] This is the communication complexity setting with shared random coins. If no random string is shared, it is called the private random coins setting. By [13] we know that the communication complexity in both the setting differs by at most a logarithmic additive factor.

One of the most well studied functions in communication complexity is the disjointness function. Given a universe $\mathcal{U}$ known to both Alice and Bob, the *disjointness function*, $\textsc{Disj}_{\mathcal{U}} : 2^{\mathcal{U}} \times 2^{\mathcal{U}} \to \{0, 1\}$, where $2^{\mathcal{U}}$ denotes the power set of $\mathcal{U}$, is defined as follows:

$$\textsc{Disj}_{\mathcal{U}}(x, y) = \begin{cases} 1, & \text{if } x \cap y = \emptyset \\ 0, & \text{o/w} \end{cases} \tag{1}$$

We also define the *intersection function*. Given a universe $\mathcal{U}$ known to both Alice and Bob, the *intersection function*, $\textsc{Int}_{\mathcal{U}} : 2^{\mathcal{U}} \times 2^{\mathcal{U}} \to 2^{\mathcal{U}}$, where $2^{\mathcal{U}}$ denotes the power set of $\mathcal{U}$, is defined as $\textsc{Int}_{\mathcal{U}}(x, y) = x \cap y$. It is easy to see that $\textsc{Int}_{\mathcal{U}}$ is harder function to compute than $\textsc{Disj}_{\mathcal{U}}$. The $\textsc{Disj}_{\mathcal{U}}$ function and its different variants, like $\textsc{Int}_{\mathcal{U}}$, have been one of the most important problems in communication complexity and have found numerous applications in areas like streaming algorithms for proving lower bounds [17, 15]. By abuse of the notation, when $\mathcal{U} = [n]$, where $[n]$ denotes the set $\{1, \ldots, n\}$, we will denote the functions $\textsc{Disj}_{[n]}$ and $\textsc{Int}_{[n]}$ by $\textsc{Disj}_n$ and $\textsc{Int}_n$ respectively.

Using the standard *rank argument* [9, 15] one can show that $D(\textsc{Disj}_n) = \Theta(n)$. In a breakthrough paper, Kalyanasundaram and Schnitger [8] proved that $R(\textsc{Disj}_n) = \Omega(n)$. Razborov [16] and Bar-Yossef et al. [1] gave alternate proofs for the above result. From the above cited results we can also see the $D(\textsc{Int}_n) = R(\textsc{Int}_n) = \Theta(n)$.

Naturally, one would also like to ask what happens to the deterministic and randomized communication complexity (one way or multiple round) of $\textsc{Disj}_n$, when both Alice and Bob know that their inputs have more structure. In particular what can we say if the inputs are guaranteed to be from a subset of $\mathcal{S} \subseteq 2^{\mathcal{U}}$, where $\mathcal{S}$ is known to both players. Let $\textsc{Disj}_{\mathcal{U}}$ functions *restricted* to $\mathcal{S} \times \mathcal{S}$ be denoted by $\textsc{Disj}_{\mathcal{U}}|_{\mathcal{S}\times\mathcal{S}}$. This problem has also been studied extensively, mostly for certain special classes of subsets $\mathcal{S} \subseteq 2^{\mathcal{U}}$. For example, the *sparse set disjointness* function, where the set $\mathcal{S}$ contains all the subsets of $\mathcal{U}$ of size at most d, is an important special case of these works.

We will denote by d-$\textsc{SparseDisj}_n$ and d-$\textsc{SparseInt}_n$, the functions $\textsc{Disj}_n|_{S\times\mathcal{S}}$ and $\textsc{Int}_n|_{S\times S}$ respectively, where S is the collections of all subsets of $[n]$ of size at most d. Using the *rank argument* [9, 15], one can again show that, for all $d \leq n$, the deterministic communication complexity of d-$\textsc{SparseDisj}_n$ is $\Omega\left(d \log \frac{n}{d}\right)$. Håstad and Wigderson [6], and Dasgupta et al. [4] showed that the randomized communication complexity and one round randomized communication complexity of d-$\textsc{SparseDisj}_n$ is $R(d\text{-}\textsc{SparseDisj}_n) = \Theta(d)$ and $R^{\rightarrow}(d\text{-}\textsc{SparseDisj}_n) = \Theta(d \log d)$ respectively. In a follow up work, Saglam and Tardos [18] proved that with $O(\log^* d)$ rounds of communication and $O(d)$ bits of communication it is possible to compute d-$\textsc{SparseDisj}_n$. More recently, Brody et al. [2] proved that $R^{\rightarrow}(d\text{-}\textsc{SparseInt}_n) = \Theta(d \log d)$ and $R(d\text{-}\textsc{SparseInt}_n) = \Theta(d)$. These results show that in the d-sparse setting, there is a separation between randomized and deterministic communication complexity of $\textsc{Disj}_n$ and $\textsc{Int}_n$ functions.

One would like to ask what happens to the communication complexity for other restrictions to the disjointness (and intersection) problem. The following are two natural problems, with a geometric flavor, for which one would like to study the communication complexity.

▶ **Problem 1** (Discrete Line Disj). Let $G \subset \mathbb{Z}^2$ be a set of n points in $\mathbb{Z}^2$ and L be the set of all lines in $\mathbb{R}^2$. Also, let $\mathcal{L} = L^d$ denote the collection of all d-size subsets of L. The Discrete Line Disj on G and $\mathcal{L}$ is a function, $\textsc{Disj}_G|_{\mathcal{L}\times\mathcal{L}}$: $\mathcal{L} \times \mathcal{L} \to \{0, 1\}$ defined as $\textsc{Disj}_G|_{\mathcal{L}\times\mathcal{L}}(\{\ell_1, \ldots, \ell_d\}, \{\ell'_1, \ldots, \ell'_d\})$ is 1 if and only if there exists a line in Alice's set[2]

[2] We assume that Alice has the set $\{\ell_1, \ldots, \ell_d\}$ and Bob has the set $\{\ell'_1, \ldots, \ell'_d\}$.

that intersects some line in Bob's set at some point in G. Formally,

$$\textsc{Disj}_G\,|_{\mathcal{L}\times\mathcal{L}}\,(\{\ell_1,\ldots,\ell_d\},\{\ell'_1,\ldots,\ell'_d\}) = \begin{cases} 1, & \text{if } \exists i,j\in[d] \text{ s.t. } \ell_i\cap\ell'_j\cap G=\emptyset \\ 0, & \text{o/w} \end{cases} \tag{2}$$

▶ **Problem 2** (Discrete Interval Disj)**.** Let $X\subset\mathbb{Z}$ be a set of n points in $\mathbb{Z}$ and Int be the set of all possible intervals. Also, let $\mathcal{I}=Int^d$ denote the collection of all d-size subsets of Int. The Discrete Interval Disj on X and $\mathcal{I}$ is a function, $\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}\colon \mathcal{I}\times\mathcal{I}\to\{0,1\}$ defined as $\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}\,(\{I_1,\ldots,I_d\},\{I'_1,\ldots,I'_d\})$ is 1 if and only if there exists an interval in Alice's set[3] that intersects some interval in Bob's set at some point in X.

$$\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}\,(\{I_1,\ldots,I_d\},\{I'_1,\ldots,I'_d\}) = \begin{cases} 1, & \text{if } \exists i,j\in[d] \text{ s.t. } I_i\cap I'_j\cap X=\emptyset \\ 0, & \text{o/w} \end{cases} \tag{3}$$

Note that both the Discrete Line Disj and Discrete Interval Disj functions are generalizations of sparse set disjointness function.[4] Although it may not be obvious at first look, but both the Discrete Line Disj function and the Discrete Interval Disj functions are disjointness functions restricted to a suitable subset. In fact, the connection between the *Sparse set disjointness* function (d-SparseDisj$_n$), the Discrete Line Disj function and the Discrete Interval Disj function run deep - all the three subsets of the domain which help to define the functions as restriction of the disjointness function have VC dimension $\Theta(d)$, see Appendix A. Naturally one would like to know, if the fact that the collection of subsets $\mathcal{S}$ has VC dimension d has any implication on the communication complexity of $\textsc{Disj}_{\mathcal{U}}\,|_{\mathcal{S}\times\mathcal{S}}$. For example, is the randomized communication complexity of Discrete Line Disj function and the Discrete Interval Disj function upper bounded by a function of d (independent of n)? And, do the Discrete Line Disj function and the Discrete Interval Disj function also have a separation between their randomized and deterministic communication complexities similar to that of the *Sparse set disjointness* function (d-SparseDisj$_n$)? We show that these are not necessarily the cases.

▶ **Theorem 3.** *For Discrete Interval Disj: there exists a $X\subset\mathbb{Z}$ with n points such that*

$$D(\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}) = D^{\to}(\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}) = R^{\to}(\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}) = \Theta\left(d\log\frac{n}{d}\right).$$

▶ **Theorem 4.** *For Discrete Line Disj: there exists a $G\subset\mathbb{Z}^2$ with n points such that $D(\textsc{Disj}_G\,|_{\mathcal{L}\times\mathcal{L}}) = D^{\to}(\textsc{Disj}_G\,|_{\mathcal{L}\times\mathcal{L}}) = \Theta\left(d\log\frac{n}{d}\right)$ and, for the randomized setting,*

$$R(\textsc{Disj}_G\,|_{\mathcal{L}\times\mathcal{L}}) = \Omega\left(d\frac{\log(n/d)}{\log\log(n/d)}\right)$$

Discrete Line Int , that is, the intersection finding version of Discrete Line Disj is defined as follows : the objective is to compute a function $\textsc{Int}_G\,|_{\mathcal{L}\times\mathcal{L}}\colon \mathcal{L}\times\mathcal{L}\to G$ that is defined as

$$\textsc{Int}_G\,|_{\mathcal{L}\times\mathcal{L}}\,(\{\ell_1,\ldots,\ell_d\},\{\ell'_1,\ldots,\ell'_d\}) = \bigcup_{i,j\in[d]}\left(\ell_i\cap\ell'_j\cap G\right).$$[5]

[3] We assume that Alice has the set $\{I_1,\ldots,I_d\}$ and Bob has the set $\{I'_1,\ldots,I'_d\}$.

[4] Take n integer points on the x-axis. For Discrete Line Disj setting, restrict only to lines orthogonal to x-axis. For Discrete Interval Disj setting, take n integer points on $\mathbb{Z}$ and only restrict to intervals containing one integer point. Both of these restriction will give the disjointness problem in the d-sparse setting.

As we have already mentioned, Brody et al. [2] proved that $R(d\text{-SparseInt}_n) = \Theta(d)$, whereas $D(d\text{-SparseInt}_n) = \Theta\left(d \log \frac{n}{d}\right)$. We show that Discrete Line Int does not demonstrate such a separation between the deterministic and randomized communication complexity.

▶ **Theorem 5.** *For Discrete Line Int : there exists a $G \subset \mathbb{Z}^2$ with n points such that*

$$D(\text{Int}_G|_{\mathcal{L}\times\mathcal{L}}) = D^{\rightarrow}(\text{Int}_G|_{\mathcal{L}\times\mathcal{L}}) = R^{\rightarrow}(\text{Int}_G|_{\mathcal{L}\times\mathcal{L}}) = R(\text{Int}_G|_{\mathcal{L}\times\mathcal{L}}) = \Theta\left(d \log \frac{n}{d}\right).$$

The upper bound for all the above three theorems can be obtained from the fact that the corresponding sets have VC dimension $\Theta(d)$, see Appendix A. *Sauer-Shelah Lemma* [19, 20, 21] states that if $\mathcal{S} \subseteq 2^{[n]}$ and VC-dim$(\mathcal{S}) \leq d$ then $|\mathcal{S}| \leq \left(\frac{en}{d}\right)^d$. Thus if VC-dim$(\mathcal{S}) \leq d$, then the Sauer-Shelah Lemma implies that $D^{\rightarrow}(\text{Int}_n|_{\mathcal{S}\times\mathcal{S}}) = O\left(d \log \frac{n}{d}\right)$. So, $O\left(d \log \frac{n}{d}\right)$ is a upper bound to the above questions, both for randomized and deterministic and also for the one-way communication. But can the randomized communication complexity of $\text{Disj}_{\mathcal{U}}|_{\mathcal{S}\times\mathcal{S}}$ and $\text{Int}_{\mathcal{U}}|_{\mathcal{S}\times\mathcal{S}}$ be even lower when S has VC dimension d? The following result, which is a direct consequence of Theorems 3, 4 and 5, enables us to we answer the question in the negative:

▶ **Theorem 6.** *Let $1 \leq d \leq n$.*

1. *There exists $\mathcal{S} \subseteq 2^{[n]}$ with VC-dim$(\mathcal{S}) \leq d$ and $R^{\rightarrow}(\text{Disj}_n|_{\mathcal{S}\times\mathcal{S}}) = \Omega\left(d \log \frac{n}{d}\right)$.*
2. *There exists $\mathcal{S} \subseteq 2^{[n]}$ with VC-dim$(\mathcal{S}) \leq d$ and $R(\text{Disj}_n|_{\mathcal{S}\times\mathcal{S}}) = \Omega\left(d \frac{\log(n/d)}{\log\log(n/d)}\right)$.*
3. *There exists $\mathcal{S} \subseteq 2^{[n]}$ with VC-dim$(\mathcal{S}) \leq d$ and $R(\text{Int}_n|_{\mathcal{S}\times\mathcal{S}}) = \Omega\left(d \log \frac{n}{d}\right)$.*

The following table compares our result with the previous best known lower bound for $\text{Disj}_{\mathcal{U}}|_{\mathcal{S}\times\mathcal{S}}$ and $\text{Int}_{\mathcal{U}}|_{\mathcal{S}\times\mathcal{S}}$ among all sets $S \subset 2^{\mathcal{U}}$ of VC dimension d.

Table 1 The largest communication complexity, for the functions $\text{Disj}_n|_{\mathcal{S}\times\mathcal{S}}$ and $\text{Int}_n|_{\mathcal{S}\times\mathcal{S}}$, among all $S \subseteq 2^{[n]}$ of VC dimension d, that was previously known and what we prove in this paper. Tight bounds of $\Omega\left(d \log \frac{n}{d}\right)$ for the largest $D(\text{Disj}_n|_{\mathcal{S}\times\mathcal{S}})$, $D^{\rightarrow}(\text{Disj}_n|_{\mathcal{S}\times\mathcal{S}})$, $D(\text{Int}_n|_{\mathcal{S}\times\mathcal{S}})$ and $D^{\rightarrow}(\text{Int}_n|_{\mathcal{S}\times\mathcal{S}})$, among all $S \subset 2^{[n]}$ of VC dimension d, follows directly from the fact that if $\mathcal{S}$ is a collection of all subsets of $[n]$ of size at most d then $D(\text{Disj}_n|_{\mathcal{S}\times\mathcal{S}}) = D(\text{Int}_n|_{\mathcal{S}\times\mathcal{S}}) = \Omega\left(d \log\left(\frac{n}{d}\right)\right)$, see [9, 15].

Problems	$R(\text{Disj}_n\|_{\mathcal{S}\times\mathcal{S}})$	$R^{\rightarrow}(\text{Disj}_n\|_{\mathcal{S}\times\mathcal{S}})$	$R(\text{Int}_n\|_{\mathcal{S}\times\mathcal{S}})$	$R^{\rightarrow}(\text{Int}_n\|_{\mathcal{S}\times\mathcal{S}})$
Previously Known	$\Omega(d)$ [6]	$\Omega(d \log d)$ [4]	$\Omega(d)$ [2]	$\Omega(d \log d)$ [2]
This Paper	$\Omega\left(d \frac{\log(n/d)}{\log\log(n/d)}\right)$	$\Omega\left(d \log \frac{n}{d}\right)$	$\Omega\left(d \log \frac{n}{d}\right)$	$\Omega\left(d \log \frac{n}{d}\right)$

Notations

We denote the set $\{1, \ldots, n\}$ by $[n]$. For a binary number $\mathbf{x}$, *decimal*$(\mathbf{x})$ denotes the decimal value of $\mathbf{x}$. For two vectors $\mathbf{x}$ and $\mathbf{y}$ in $\{0,1\}^n$, $\mathbf{x} \cap \mathbf{y} = \{i \in [n] : \mathbf{x}_i = \mathbf{y}_i = 1\}$, and $\mathbf{x} \subseteq \mathbf{y}$ when $\mathbf{x}_i \leq \mathbf{y}_i$ for each $i \in [n]$. For a finite set X, 2^X denotes the power set of X. For $x, y \in \mathbb{R}$ with $x < y$, $[x, y]$ denotes the closed interval is the set of all real numbers that lies between x and y.

2 One way communication complexity (Theorems 3 and 6 (1))

In this section we will prove the following result.

▶ **Theorem 7.** *For all $n \geq d$, there exists $X \subset \mathbb{Z}$ with $|X| = n$ and $\mathcal{R} \subseteq 2^X$ with VC-dim$(\mathcal{R}) = 2d$, such that*

$$\mathcal{R} \subseteq \left\{ X \cap \left(\bigcup_{1 \leq j \leq d} I_j \right) \mid \{I_1, \ldots, I_d\} \in \mathcal{I} \right\} \text{ and } R^{\rightarrow}(\textsc{Disj}_X \mid_{\mathcal{R} \times \mathcal{R}}) = \Omega\left(d \log \frac{n}{d}\right).$$

Note that the set $\mathcal{I}$ is defined in Problem 2.

▶ Remark 8. The above result takes care of the proofs of Theorem 3 and Theorem 6 (1).

The *hard* instance, for the proof of the above theorem, is inspired by the *interval* set systems in combinatorial geometry and is constructed in Section 2.1. In Section 2.2, we proof Theorem 7 by using a reduction from Augmented Indexing, which we denote by AugIndex$_\ell$. Formally the problem AugIndex$_\ell$ is defined as follows: Alice gets a string $\mathbf{x} \in \{0,1\}^\ell$ and Bob gets an index $j \in [\ell]$ and all $x_{j'}$ for $j' < j$. Bob reports x_j as the output.

▶ **Proposition 9.** (Miltersen et al. [12]) $R^{\rightarrow}(\textsc{AugIndex}_\ell) = \Omega(\ell)$.

2.1 Construction of a hard instance

We construct a set $X \subset \mathbb{Z}$ with $|X| = n$ and $\mathcal{R} \subseteq 2^X$ with VC-dim$(\mathcal{R}) = 2d$. Informally, X is the union of the set of points present in the union of d pairwise disjoint intervals, in $\mathbb{Z}$, each containing $\frac{n}{d}$ points. Each set in $\mathcal{R}$ is the union of the set of points in the subintervals anchored either at the left or the right end point of each of the above d intervals. Formally, the description of X and $\mathcal{R}$ are given below along with some of its properties that are desired to show Theorem 7.

The ground set X

Let $m = \frac{n}{d} - 2$. Without loss of generality we can assume that $m = 2^k$, where $k \in \mathbb{N}$. Let $J_0 = \{0, \ldots, m+1\}$ be the set of $m+2$ consecutive integers that starts from the origin and ends at $m+1$. Similarly, let J_p be the set of $m+1$ consecutive integers that starts at $p \in \mathbb{Z}$ and ends at $p+m+1$. Let $p_1, \ldots, p_d$ be d points in $\mathbb{Z}$ such that the sets $J_{p_1}, \ldots, J_{p_d}$ are pairwise disjoint. Let the *ground* set X be $\bigcup_{i=1}^{d} J_{p_i}$. Note that $X \subset \mathbb{Z}$ and $|X| = (m+2)d = n$.

The subsets of X in $\mathcal{R}$

$\mathcal{R} \subseteq 2^X$ contains two types of sets $\mathcal{R}_0$ and $\mathcal{R}_{m+1}$, where

- Take any d intervals $R_1, \ldots, R_d$ of integer lengths such that, for all $i \in [d]$, length of R_i is at most $m+1$, $R_i \subseteq [p_i, p_i + m + 1]$, and R_i starts at p_i. Note that R_i does not intersect with any $X \setminus J_{p_i}$. The set $A = \bigcup_{i=1}^{d} (R_i \cap X)$ is an element in $\mathcal{R}_0$. We say that A is *generated* by $R_1, \ldots, R_d$.
- Take any d intervals $R'_1, \ldots, R'_d$ of integer lengths such that, for all $i \in [d]$, length of R'_i is at most $m+1$, $R'_i \subseteq [p_i, p_i + m + 1]$ and R'_i ends at $p_i + m + 1$. Note that R'_i does not intersect with any $X \setminus J_{p_i}$. The set $B = \bigcup_{i=1}^{d} (R'_i \cap X)$ is an element in $\mathcal{R}_{m+1}$. We say that B is *generated* by $R'_1, \ldots, R'_d$.

The following claim bounds the VC dimension of $\mathcal{R}$, constructed as above.

▷ Claim 10. For $X \subset \mathbb{Z}$ with $|X| = n$ and $\mathcal{R} \subset 2^X$ as described above, VC-dim$(\mathcal{R}) = 2d$,

Proof. The proof follows from the fact that any subset of of X containing $2d+1$ points will contain at least three points from some J_{p_i}, where $i \in [d]$. These points in J_{p_i} can not be shattered by the sets in $\mathcal{R}$. Also, observe that there exists $2d$ points, with two from each J_{p_j}, that can be shattered by the sets in $\mathcal{R}$. ◁

Now, we give a claim about X and $\mathcal{R}$ constructed above that will be required for our proof of Theorem 7.

▷ Claim 11. Let $A \in \mathcal{R}_0$ and $B \in \mathcal{R}_{m+1}$ be such that A is generated by $R_1, \ldots, R_d$ and B is generated by $R'_1, \ldots, R'_d$. Then A and B intersects if and only if there exists an $i \in [d]$ such that R_i intersects R'_i at a point in J_{p_i}.

The proof of Claim 11 follows directly from our construction of $X \subset \mathbb{Z}$ and $\mathcal{R} \subseteq 2^X$, as $J_{p_1}, \ldots, J_{p_d}$ are pairwise disjoint.

2.2 Reduction from AugIndex$_{d \log m}$ to Disj$_X$ $|_{\mathcal{R}\times\mathcal{R}}$

Before presenting the reduction we recall the definitions of AugIndex$_{d\log m}$ and Disj$_X|_{\mathcal{R}\times\mathcal{R}}$. In AugIndex$_{d\log m}$, Alice gets $\mathbf{x} \in \{0,1\}^{d\log m}$ and Bob gets an index j and $x_{j'}$ for each $j' < j$. The objective of Bob is to report x_j as the output. In Disj$_X|_{\mathcal{R}\times\mathcal{R}}$, Alice gets $A \in \mathcal{R}_0$ and Bob gets $B \in \mathcal{R}_{m+1}$. The objective of Bob is to determine whether $A \cap B = \emptyset$. Note that $X, \mathcal{R}, \mathcal{R}_0$ and $\mathcal{R}_{m+1}$ are as discussed in the Section 2.1.

Let $\mathcal{P}$ be an one-way protocol that solves Disj$_X|_{\mathcal{R}\times\mathcal{R}}$ with $o\left(d \log \frac{n}{d}\right) = o(d \log m)$ bits of communication. Now, we consider the following protocol $\mathcal{P}'$ for AugIndex$_{d\log m}$ that has the same one way communication cost as that of Disj$_X|_{\mathcal{R}\times\mathcal{R}}$. Then we will be done with the proof of Theorem 7.

Protocol $\mathcal{P}'$ for AugIndex$_{d\log m}$ problem

Step-1 Let $\mathbf{x} \in \{0,1\}^{d\log m}$ be the input of Alice. Bob gets an index $j \in [d \log m]$ and bits $x_{j'}$ for each $j' < j$.

Step-2 Alice will form d strings $\mathbf{a}_1, \ldots, \mathbf{a}_d \in \{0,1\}^{\log m}$ by partitioning the string $\mathbf{x}$ into d parts such that $\mathbf{a}_i = x_{(i-1)\log m+1} \ldots x_{i \log m}$, where $i \in [d]$. Bob first forms a string $\mathbf{y} \in \{0,1\}^{d\log m}$, where $y_{j'} = x_{j'}$ for each $j' < j$, $y_j = 1$, and $y_{j'} = 0$ for each $j' > j$. Then Bob finds $\mathbf{b}_1, \ldots, \mathbf{b}_d \in \{0,1\}^{\log m}$ by partitioning the string $\mathbf{y}$ in to d parts such that $\mathbf{b}_i = y_{(i-1)\log m+1} \ldots y_{i \log m}$, where $i \in [d]$.

Step-3 For each $i \in [d]$, let R_i and R'_i be the intervals that starts at p_i and ends at $p_i + m + 1$, respectively, where $R_i = [p_i, m + p_i - \mathit{decimal}(\mathbf{a}_i)]$ and $R'_i = [p_i + m + 1 - \mathit{decimal}(\mathbf{b}_i), p_i + m + 1]$. Alice finds the set $A \in \mathcal{R}_0$ generated by $R_1, \ldots, R_d$ and Bob finds the set $B \in \mathcal{R}_{m+1}$ generated by $R'_1, \ldots, R'_d$, i.e., $A = \bigcup_{i\in[d]} (R_i \cap X)$ and $B = \bigcup_{i\in[d]} (R'_i \cap X)$.

Step-4 Alice and Bob solves Disj$_X|_{\mathcal{R}\times\mathcal{R}}$ on inputs A and B, and report $x_j = 0$ if and only if Disj$_X|_{\mathcal{R}\times\mathcal{R}}(A, B) = 0$. Note that x_j is the output of AugIndex$_{d\log m}$ problem.

The following observation follows from the description of the protocol $\mathcal{P}'$ and from the construction of $X \subset \mathbb{Z}$ and $\mathcal{R} \subseteq 2^X$.

▶ **Observation 12.** Let $i^* \in [d]$ such that $j \in \{(i^*-1)\log m + 1, i^* \log m\}$. Then
(i) $R_i \cap R'_i = \emptyset$ for all $i \neq i^*$.
(ii) $R_{i^*} \cap R'_{i^*} = \emptyset$ if and only if $\mathit{decimal}(\mathbf{b}_{i^*}) \leq \mathit{decimal}(\mathbf{a}_{i^*})$.
(iii) $\mathit{decimal}(\mathbf{b}_{i^*}) \leq \mathit{decimal}(\mathbf{a}_{i^*})$ if and only if $x_j = 0$.
We will use the above observation to show the correctness of the protocol $\mathcal{P}'$.

First consider the case $\textsc{Disj}_X\ |_{\mathcal{R}\times\mathcal{R}}\ (A, B) = 0$. Then, by Claim 11, there exists an $i \in [d]$ such that R_i and R'_i intersects at a point in J_{p_i}. From Observation 12 (i), we can say $R_{i*} \cap R'_{i*} \neq \emptyset$. Combining $R_{i*} \cap R'_{i*} \neq \emptyset$ with Observations 12 (ii) and (iii), we have $x_j = 0$. Hence, $\textsc{Disj}_X\ |_{\mathcal{R}\times\mathcal{R}}\ (A, B) = 0$ implies $x_j = 0$. The converse part, i.e., $x_j = 0$ implies $\textsc{Disj}_X\ |_{\mathcal{R}\times\mathcal{R}}\ (A, B) = 0$, can be shown in the similar fashion.

The one-way communication complexity of protocol $\mathcal{P}'$ for $\textsc{AugIndex}_{d\log m}$ is the same as that of $\mathcal{P}$ for $\textsc{Disj}_X|_{\mathcal{R}\times\mathcal{R}}$, that is, $o(d\log m)$. However, this is impossible as the one-way communication complexity of Augmented Indexing, over $d\log m$ bits, is $\Omega(d\log m) = \Omega\left(d\log\frac{n}{d}\right)$ bits. This completes the proof of Theorem 7.

3 Two way communication complexity (Theorems 4, 5, 6 (2) and 6 (3))

In this section, we prove the following theorems.

▶ **Theorem 13.** *For all $n \geq d$, there exists a $G \subset \mathbb{Z}^2$ with $|G| = n$ and $\mathcal{T} \subseteq 2^G$ with VC-dim$(\mathcal{T}) = 2d$, such that*

$$\mathcal{T} \subseteq \left\{ G \cap \left(\bigcup_{1\leq j\leq d} \ell_j \right) \mid \{\ell_1, \ldots, \ell_d\} \in \mathcal{L} \right\} \text{ and } R(\textsc{Disj}_G\ |_{\mathcal{T}\times\mathcal{T}}) = \Omega\left(d\frac{\log(n/d)}{\log\log(n/d)} \right).$$

The set $\mathcal{L}$ is as defined in Problem 1.

▶ **Theorem 14.** *For all $n \geq d$, there exists a $G \subset \mathbb{Z}^2$ with $|G| = n$ and $\mathcal{T} \subseteq 2^G$ with VC-dim$(\mathcal{T}) = 2d$, such that*

$$\mathcal{T} \subseteq \left\{ G \cap \left(\bigcup_{1\leq j\leq d} \ell_j \right) \mid \{\ell_1, \ldots, \ell_d\} \in \mathcal{L} \right\} \text{ and } R(\textsc{Int}_G\ |_{\mathcal{T}\times\mathcal{T}}) = \Omega\left(d\log\frac{n}{d} \right).$$

The set $\mathcal{L}$ is as defined in problem 1.

▶ Remark 15. Theorem 13 takes care of Theorem 4 and 6(2). Theorem 14 takes care of Theorem 5 and 6(3).

Note that the same set system will be used for the proofs of the above theorems. The *hard* instance, for the proof of the above theorems, is inspired by *point line incidence* set systems in computational geometry and is constructed in Section 3.1. We prove Theorems 13 and 14 in Sections 3.2 and 3.3, respectively, using reductions.

3.1 The hard instance for the proofs of Theorems 13 and 14

In this subsection, we give the description of $G \subset \mathbb{Z}^2$ with $|G| = n$ and $\mathcal{T} \subseteq 2^G$, with VC-Dim$(\mathcal{T}) = 2d$. The same G and $\mathcal{T}$ will be our *hard* instance for the proofs of Theorems 13 and 14. In this subsection, without loss of generality, we can assume that d divides n and n/d is a perfect square.

Informally, G is the set of points present in the union of d many pairwise disjoint square grids each containing $\frac{n}{d}$ points and the grids are taken in such a way that any straight line of non-negative can intersects with at most one grid. Also, each set in $\mathcal{T}$ is the union of the set of points present in d many lines of non-negative slope such that one line intersects with exactly one grid. Moreover, all of the d lines have slopes either zero or positive. Formally, the description of G and $\mathcal{T}$ are given below along with some of its properties that are desired to show Theorems 13 and 14.

The ground set G

Let $m = \sqrt{\frac{n}{d}}$, and $G_{(0,0)} = \{(x, y) \in \mathbb{Z}^2 : 0 \leq x, y \leq m-1\}$ be the grid of size $m \times m$ anchored at the origin $(0, 0)$. For any $p, q \in \mathbb{Z}$, the $m \times m$ grid anchored at (p, q) will be denoted by $G_{(p,q)}$, i.e., $G_{(p,q)} = \{(i+p, j+q) : (i, j) \in G_{(0,0)}\}$. For $d \in \mathbb{N}$, consider $G_{(p_1,q_1)}, \ldots, G_{(p_d,q_d)}$ satisfying the following property:

PROPERTY *For any $i, j \in [d]$, with $i \neq j$, let L_1 and L_2 be lines of non-negative slopes that pass through at least two points of $G_{(p_i,q_i)}$ and $G_{(p_j,q_j)}$, respectively. Then L_1 and L_2 does not intersect at any point* inside $\bigcup_{\ell=1}^{d} G_{(p_\ell,q_\ell)}$.

Observe that there exists $G_{(p_1,q_1)}, \ldots, G_{(p_d,q_d)}$ satisfying PROPERTY. We will take $G = \bigcup_{\ell=1}^{d} G_{(p_\ell,q_\ell)}$ as the ground set. Without loss of generality, we can assume that $(p_1, q_1) = (0, 0)$. Note that $G \subset \mathbb{Z}^2$ and $|G| = dm^2 = n$.

The subsets of G in $\mathcal{T}$

$\mathcal{T}$ contains two types of subsets $\mathcal{T}_1$ and $\mathcal{T}_2$ of G, and they are generated by the following ways:

- Take any d lines $L_1, \ldots, L_d$ of non negative slope such that, $\forall i \in [d]$, L_i passes through $(p_i, q_i) \in G_{(p_i,q_i)}$ and (at least) another point in $G_{(p_i,q_i)}$. Note that L_i does not contain any point from $G \setminus G_{(p_i,q_i)}$. The set $A = \bigcup_{i=1}^{d} \left(L_i \cap G_{(p_i,q_i)}\right)$ is in $\mathcal{T}_1$, and we say A is *generated* by the lines $L_1, \ldots, L_d$.
- Take any d vertical lines $L'_1, \ldots, L'_d$ such that, $\forall i \in [d]$, L'_i contains at least one point from $G_{(p_i,q_i)}$. Note that L'_i does not contain any point from $G \setminus G_{(p_i,q_i)}$. The set $B = \bigcup_{i=1}^{d} (L'_i \cap G_{(p_i,q_i)})$ is in $\mathcal{T}_2$, and we say B is generated by the lines $L'_1, \ldots, L'_d$.

The following claim bounds the VC dimension of $\mathcal{T}$, which as described above.

▷ Claim 16. For $G \subset \mathbb{Z}^2$ and $\mathcal{T} \subseteq 2^G$ as described above, VC-dim$(\mathcal{T}) = 2d$.

Proof. The proof follows from the fact that any subset of X containing $2d + 1$ points will contain at least three points from some $G_{(p_j,q_j)}, j \in [d]$. These points in $G_{(p_j,q_j)}$ can not be shattered by the sets in $\mathcal{T}$. Also, observe that there exists $2d$ points two from each $G_{(p_j,q_j)}$ that can be shattered by the sets in $\mathcal{T}$. ◁

Now, we give two claims about G and $\mathcal{T}$, constructed above, that follow directly from our construction of $G \subset \mathbb{Z}^2$ and $\mathcal{T} \subseteq 2^G$.

▷ Claim 17. Let $A \in \mathcal{T}_1$ and $B \in \mathcal{T}_2$ such that A is generated by lines $L_1, \ldots, L_d$ and A is generated by lines $L'_1, \ldots, L'_d$. Then A and B intersect if and only if there exists $i \in [d]$ such that L_i and L'_i intersect at a point in $G_{(p_i,q_i)}$.

▷ Claim 18. Let $A \in \mathcal{T}_1$ and $B \in \mathcal{T}_2$ such that A is generated by lines $L_1, \ldots, L_d$ and B is generated by lines $L'_1, \ldots, L'_d$. Also let $|A \cap B| = d$. Then for each $i \in [d]$, L_i and L'_i intersect at a point in $G_{(p_i,q_i)}$. Moreover, A (B) can be determined if we know B (A) and $A \cap B$.

The above claims will be used in the proofs of Theorems 13 and 14.

3.2 Proof of Theorem 13

Let us consider a problem in communication complexity denoted by Or-Disj$^t_{\{0,1\}^\ell}$ that will be used in our proof. In Or-Disj$^t_{\{0,1\}^\ell}$, Alice gets t strings $\mathbf{x}_1, \ldots, \mathbf{x}_t \in \{0,1\}^\ell$ and Bob also gets t strings $\mathbf{y}_1, \ldots, \mathbf{y}_t \in \{0,1\}^\ell$. The objective is to compute $\bigvee_{i=1}^{t} \textsc{Disj}_{\{0,1\}^\ell}(\mathbf{x}_i, \mathbf{y}_i)$. Note that $\textsc{Disj}_{\{0,1\}^\ell}(\mathbf{x}_i, \mathbf{y}_i)$ is a binary variable that takes value 1 if and only if $\mathbf{x}_i \cap \mathbf{y}_i = \emptyset$.

▶ **Proposition 19** (Jayram et al. [7]). $R\left(\textsc{Or-Disj}^t_{\{0,1\}^\ell}\right) = \Omega(\ell t)$.

Note that Proposition 19 directly implies the following result.

▶ **Proposition 20.** $R\left(\textsc{Or-Disj}^t_{\{0,1\}^\ell} \mid_{S_\ell \times S_\ell}\right) = \Omega(\ell t)$, *where* $S_\ell = \{0,1\}^\ell \setminus \{0^\ell\}$.

Let $k \in \mathbb{N}$ be the largest integer such that first k consecutive primes $p_1, \ldots, p_k$ satisfy the following inequalty:

$$\Pi_{i=1}^{k} p_i \leq \sqrt{\frac{n}{d}}. \tag{4}$$

Using the fact that $\Pi_{i=1}^{k} p_i = e^{(1+o(1))k \log k}$, we get $k = \Theta\left(\frac{\log(n/d)}{\log\log(n/d)}\right)$.

We prove the theorem by a reduction from Or-Disj$^d_{\{0,1\}^k} \mid_{S_k \times S_k}$ to $\textsc{Disj}_G \mid_{\mathcal{T}\times\mathcal{T}}$, where $S_k := \{0,1\}^k \setminus \{0^k\}$.

Note that $G \subset \mathbb{Z}^2$ with $|G| = n$, and $\mathcal{T} \subseteq 2^G$, with VC-dim$(\mathcal{T}) = 2d$, are the same as that we constructed in Section 3.1. To reach a contradiction, assume that there exists a two way protocol $\mathcal{P}$ that solves $\textsc{Disj}_G \mid_{\mathcal{T}\times\mathcal{T}}$ with communication cost of $o\left(d\frac{\log m}{\log\log m}\right) = o\left(d\frac{\log(n/d)}{\log\log(n/d)}\right)$ bits. Now, we give protocol $\mathcal{P}'$ that solves Or-Disj$^d_{\{0,1\}^k} \mid_{S_k \times S_k}$, as described below.

Protocol $\mathcal{P}'$ for Or-Disj$^d_{\{0,1\}^k} \mid_{S_k \times S_k}$

Step-1 Let $A = (\mathbf{x}_1, \ldots, \mathbf{x}_d) \in [S_k]^d$ [6] and $B = (\mathbf{y}_1, \ldots, \mathbf{y}_d) \in [S_k]^d$ be the inputs of Alice and Bob for Or-Disj$^d_{\{0,1\}^k} \mid_{S_k \times S_k}$. Recall that $S_k = \{0,1\}^k \setminus \{0^k\}$. Bob finds $\bar{B} = (\bar{\mathbf{y}}_1, \ldots, \bar{\mathbf{y}}_d) \in \left[\{0,1\}^k\right]^d$, where $\bar{\mathbf{y}}_i$ is obtained by complementing each bit of $\mathbf{y}_i$.

Step-2 Both Alice and Bob privately determine first k prime numbers $p_1, \ldots, p_k$ without any communication.

Step-3 Let $\Phi : \{0,1\}^k \to \{0,1\}^{\lceil \log(\sqrt{\frac{n}{d}}) \rceil}$ be the function such that $\phi(\mathbf{x})$ is the bit representation of the number $\prod_{i=1}^{k} p_i^{x_i}$, where $\mathbf{x} = (x_1, \ldots, x_k) \in \{0,1\}^k$. Alice finds $A' = (\mathbf{a}_1, \ldots, \mathbf{a}_d) \in \left[\{0,1\}^{\lceil \log(\sqrt{\frac{n}{d}}) \rceil}\right]^d$ and Bob finds $B' = (\mathbf{b_1}, \ldots, \mathbf{b_1} \in \left[\{0,1\}^{\lceil \log(\sqrt{\frac{n}{d}}) \rceil}\right]^d$ privately without any communication, where $\mathbf{a_i} = \phi(\mathbf{x_i})$ and $\mathbf{b_i} = \phi(\bar{\mathbf{y}}_i)$ for each $i \in [d]$.

[6] For a set W, $[W]^d = W \times \ldots \times W$ (d times).

Step-4 For each $i \in [d]$, let L_i and L'_i be the lines having equation $y - q_i = \frac{decimal(\mathbf{a}_i)-1}{decimal(\mathbf{a}_i)}(x - p_i)$ and $x - p_i = decimal(\mathbf{b}_i)$ respectively. Alice finds $A'' \in \mathcal{T}$ that is generated by the lines $L_1, \ldots, L_d$, and Bob finds $B'' \in \mathcal{T}$ which is generated by the lines $L'_1, \ldots, L'_d$, i.e., $A'' = \bigcup_{i\in[d]} (L_i \cap G_{(p_i,q_i)})$ and $B'' = \bigcup_{i\in[d]} (L'_i \cap G_{(p_i,q_i)})$.

Step-5 Then Alice and Bob solve $\textsc{Disj}_G\,|_{\mathcal{T}\times\mathcal{T}}\,(A'', B'')$, and report $\bigvee_{i=1}^{d} \textsc{Disj}_{\{0,1\}^k}(\mathbf{x}_i, \mathbf{y}_i) = 1$ if and only if $\textsc{Disj}_G\,|_{\mathcal{T}\times\mathcal{T}}\,(A'', B'') = 0$.

Now we argue for the correctness of the protocol $\mathcal{P}'$. Let $\textsc{Disj}_G\,|_{\mathcal{T}\times\mathcal{T}}\,(A'', B'') = 0$, that is, $A'' \cap B'' \neq \emptyset$. By Claim 17 and from the description of $\mathcal{P}'$, there exists $i \in [d]$ such that the lines $L_i : y - q_i = \frac{decimal(\mathbf{a}_i)-1}{decimal(\mathbf{a}_i)}(x - p_i)$ and $L'_i : x - p_i = decimal(\mathbf{b_i})$ intersect at a point in $G_{(p_i,q_i)}$, that is, the lines $y = \frac{decimal(\mathbf{a}_i)-1}{decimal(\mathbf{a}_i)}x$ and $x = decimal(\mathbf{b}_i)$ intersect at a point in $G_{(0,0)}$. Now, we can say that, there exists $i \in [d]$ such that $decimal(\mathbf{a}_i)$ divides $decimal(\mathbf{b_i})$, equivalently, $\phi(\mathbf{x}_i)$ divides $\phi(\bar{\mathbf{y}}_i)$. This implies $\mathbf{x}_i$ is a subset of $\bar{\mathbf{y}}_i$ (or $\mathbf{x}_i \cap \mathbf{y_i} = \emptyset$) for some $i \in [d]$. Hence, $\bigvee_{i=1}^{d} \textsc{Disj}_{\{0,1\}^k}(\mathbf{x}_i, \mathbf{y}_i) = 1$. The converse part, that is, $\bigvee_{i=1}^{d} \textsc{Disj}_{\{0,1\}^k}(\mathbf{x}_i, \mathbf{y}_i) = 1$ implies $\textsc{Disj}_G\,|_{\mathcal{T}\times\mathcal{T}}\,(A'', B'') = 0$ can be shown in the similar fashion.

Observe that the communication cost of protocol $\mathcal{P}'$ for $\textsc{Or-Disj}^d_{\{0,1\}^k}\,|_{S\times S}$ is same as that of protocol $\mathcal{P}$ for $\textsc{Disj}_G\,|_{\mathcal{T}\times\mathcal{T}}$, which is $o\left(d\frac{\log m}{\log\log m}\right) = o\left(d\frac{\log(n/d)}{\log\log(n/d)}\right) = o(dk)$ as $m = \sqrt{\frac{n}{d}}$ and $k = \Theta\left(\frac{\log(n/d)}{\log\log(n/d)}\right)$. This contradicts Proposition 20 which says that $R\left(\textsc{Or-Disj}^d_{\{0,1\}^k}\,|_{S\times S}\right) = \Omega(dk)$.

3.3 Proof of Theorem 14

With out loss of generality, we also assume that d divides n and, more over, n/d is a perfect square.

First, consider the problem $\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}$, where the objective of Alice and Bob is to learn each other's set. Note that $G \subset \mathbb{Z}^2$ with $|G| = n$ and $\mathcal{T} \subseteq 2^G$ with VC-Dim$(\mathcal{T}) = 2d$ are same as that constructed in Section 3.1. In $\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}$, Alice and Bob get two sets A and B, respectively, from $\mathcal{T}$ with a promise $|A \cap B| = d$. The objective of Alice (Bob) is to learn B (A). Observe that $R(\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}) = \Omega(d\log n)$ as there are $\Omega(m^d) = \Omega\left(\left(\sqrt{\frac{n}{d}}\right)^d\right)$ many candidate sets for the inputs of Alice and Bob. We prove the theorem by a reduction from $\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}$ to $\textsc{Int}_G\,|_{\mathcal{T}\times\mathcal{T}}$.

Let by contradiction consider a protocol $\mathcal{P}$ that solves $\textsc{Int}_G\,|_{\mathcal{T}\times\mathcal{T}}$ by using $o(d\log n)$ bits of communication. To solve $\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}$, Alice and Bob first run a protocol $\mathcal{P}$ and finds $A \cap B$. Now by Claim 17, it is possible for Alice (Bob) to determine B (A) by combining A (B) along with $A \cap B$, with out any communication with Bob (Alice). Now, we have a protocol $\mathcal{P}'$ that solves $\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}$ with $o(d\log n)$ bits of communication. However, this is impossible as $R(\textsc{Learn}_G\,|_{\mathcal{T}\times\mathcal{T}}) = \Omega(d\log n)$. Hence, we are done with the proof of Theorem 14.

4 Conclusion and Discussion

In this paper, we studied $\textsc{Disj}_n\,|_{\mathcal{S}\times\mathcal{S}}$ and $\textsc{Int}_n\,|_{\mathcal{S}\times\mathcal{S}}$ when $\mathcal{S}$ is a subset of $2^{[n]}$ and VC-dim$(\mathcal{S}) \leq d$. One of the main contributions of our work is the result (Theorem 6) showing that unlike in the case of d-$\textsc{SparseDisj}_n$ and d-$\textsc{SparseDisj}_n$ functions, there is no separation between randomized and deterministic communication complexity of $\textsc{Disj}_n\,|_{\mathcal{S}\times\mathcal{S}}$ and

$\textsc{Int}_n|_{\mathcal{S}\times\mathcal{S}}$ functions when VC-dim$(S) \leq d$. Note that we have settled both the one-way and two-way (randomized) communication complexities of $\textsc{Int}_n|_{\mathcal{S}\times\mathcal{S}}$ when VC-dim$(\mathcal{S}) \leq d$ (Theorem 6 (1) and (3)). In the context of $\textsc{Disj}_n|_{\mathcal{S}\times\mathcal{S}}$, we have settled the one-way (randomized) communication complexity. The two-way communication complexity for $\textsc{Disj}_n|_{\mathcal{S}\times\mathcal{S}}$ is tight up to factor $\log\log\frac{n}{d}$ (See Theorem 6 (2)). However, we believe that the factor of $\log\log\frac{n}{d}$ should not be present in the statement of Theorem 6 (2).

▶ **Conjecture 21.** *There exists $\mathcal{S} \subseteq 2^{[n]}$ with VC-dim$(\mathcal{S}) \leq d$ and $R(\textsc{Disj}_n|_{\mathcal{S}\times\mathcal{S}}) = \Omega\left(d\log\frac{n}{d}\right)$.*

Recall $G \subset \mathbb{Z}^2$ with $|G| = n$ and $\mathcal{T} \subseteq 2^G$ with VC-Dim$(\mathcal{T}) = 2d$ construction from Section 3.1, that served as the hard instance for the proof of Theorem 13 and Theorem 14. The same G and $\mathcal{T}$ can not be the hard instance for the proof of Conjecture 21 because of the following result.

▶ **Theorem 22.** *Let us consider $G \subset \mathbb{Z}^2$ with $|G| = n$ and $\mathcal{T} \subseteq 2^G$ with VC-Dim$(\mathcal{T}) = 2d$ as defined in Section 3.1. Also, recall the definition of $\mathcal{T}_1$ and $\mathcal{T}_2$. There exists a randomized communication protocol that can, $\forall A \in \mathcal{T}_1$ and $\forall B \in \mathcal{T}_2$, can compute $\textsc{Disj}_G|_{\mathcal{T}\times\mathcal{T}}(A,B)$, with probability at least $2/3$, and uses $O\left(\frac{d\log d\log\frac{n}{d}}{\log\log\frac{n}{d}}\cdot\log\log\log\frac{n}{d}\right)$ bits of communication.*

We use the following observation to prove the above theorem.

▶ **Observation 23.** Let us consider the communication problem $\textsc{Gcd}_k(a,b)$, where Alice and Bob get a and b respectively from $\{1,\ldots,k\}$, and the objective is for both the players to compute $\gcd(a,d)$. Then there exists a randomized protocol, with success probability at least $1-\delta$, for $\textsc{Gcd}_k$ that uses $O\left(\frac{\log k}{\log\log k}\cdot\log\log\log k\cdot\log\frac{1}{\delta}\right)$ bits of communication.

Proof. We will give a protocol P for the case when $\delta = 1/3$ that uses $O\left(\frac{\log k}{\log\log k}\cdot\log\log\log k\right)$ bits of communication. By repeating $O\left(\log\frac{1}{\delta}\right)$ times protocol $\mathcal{P}$ and reporting the majority of the outcomes as the output, we will get the correct answer with probability at least $1-\delta$. Both Alice and Bob generate all the prime numbers $p_1, \ldots, p_t$ between 1 and k. From the Prime Number Theorem, we known that $t = \Theta\left(\frac{k}{\log k}\right)$. Alice and Bob separately, construct the sets S_a and S_b that contain the prime numbers that divides a and b respectively. Note that $|S_a|$ and $|S_b|$ is bounded by $O\left(\frac{\log k}{\log\log k}\right)$.[7] Alice and Bob compute $S_a \cap S_b$ by solving *Sparse Set Intersection* problem on input S_a and S_b using $O\left(\frac{\log k}{\log\log k}\right)$ bits of communication [2]. For $p \in S_a \cap S_b$, let $\alpha_{p,a}$ and $\alpha_{p,b}$ denote the exponent of p in a and b, respectively. Observe that

$$\gcd(a,b) = \prod_{p\in S_a\cap S_b} p^{\min\{\alpha_{p,a},\alpha_{p,b}\}}.$$

For each $p \in S_a$, Alice sends $\alpha_{p,a}$ to Bob. Number of bits of communication required to send the exponents of all the primes in $S_a \cap S_b$, is

[7] The product of first t prime numbers is $e^{(1+o(1))t\log t}$.

$$|S_a \cap S_b| + \sum_{p \in S_a \cap S_b} \log(\alpha_{p,a}) \leq O\left(\frac{\log k}{\log\log k}\right) + |S_a \cap S_b| \log\left(\frac{\sum_{p \in S_a \cap S_b} \alpha_{p,a}}{|S_a \cap S_b|}\right)$$
$$\leq O\left(\frac{\log k}{\log\log k}\right) + |S_a \cap S_b| \log\left(\frac{\log k}{|S_a \cap S_b|}\right)$$
$$\leq O\left(\frac{\log k}{\log\log k} \cdot \log\log\log k\right)$$

In the above inequalities, we used the facts that $|S_a \cap S_b| = O\left(\frac{\log k}{\log\log k}\right)$, $\sum_{p \in S_a \cap S_b} \alpha_{p,a} \leq \log k$ and $\log x$ is a concave function. After getting the exponents $\alpha_{p,a}$ of the primes $p \in S_a \cap S_b$ from Alice, Bob also sends the exponents $\alpha_{p,b}$ of the primes $p \in S_a \cap S_b$ to Alice using $O\left(\frac{\log k}{\log\log k} \log\log\log k\right)$ bits of communication to Alice. Since both Alice and Bob now know the set $S_a \cap S_b$, and the exponents $\alpha_{p,a}$ and $\alpha_{p,b}$ for all $p \in S_a \cap S_b$, both of them can compute $\gcd(a, b)$. Total number of bits communicated in this protocol is $O\left(\frac{\log k}{\log\log k} \log\log\log k\right)$. ◀

We will now give the proof of Theorem 22.

Proof of Theorem 22. Consider the case when $d = 1$. From the description of G and $\mathcal{T}$ in Section 3.1, we can say that $G = G_{(0,0)}$, where $G_{(0,0)} = \{(x, y) \in \mathbb{Z}^2 : 0 \leq x, y \leq \sqrt{n}\}$ [8]. Moreover, each set in $\mathcal{T}_1$ is a set of points present in a straight line of non-negative slope that passes through two points of $G_{(0,0)}$ with one point being $(0, 0)$ and each set in $\mathcal{T}_2$ is a set of points present in a vertical straight line that passes through exactly $\sqrt{n}$ many grid points. Keeping Claims 17 and 18 in mind, we will be done if we can show the existence of a randomized communication protocol for computing the function $\textsc{Disj}_G\ |_{\mathcal{T} \times \mathcal{T}}$, with probability of success at least $1 - \delta$ and number of bits communicated by the protocol being bounded by $O\left(\frac{\log n}{\log\log n} \cdot \log\log\log n \cdot \log\frac{1}{\delta}\right)$, for the special case when $d = 1$. This is because for general d, we will be solving d instances of the above problem, with the number of points in each grid being $\frac{n}{d}$ [9] and setting $\delta = \frac{1}{3d}$ for each of the d instances.

Protocol for $d = 1$

Alice and Bob get A and B from $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. Let A is generated by the straight line L_A and B is generated by L_B, where L_A is a straight line with non-negative slope and L_B is a vertical line. If L_A is a horizontal one : Alice just sends this information to Bob and then both report that $A \cap B \neq \emptyset$. If L_A is a vertical line : Alice sends this information to Bob and he reports $A \cap B \neq \emptyset$ if and only if L_B passes through origin. Now assume that L_A is neither a horizontal nor a vertical line. Let the equation of L_A be $y = \frac{p}{q}x$, where $1 \leq p, q \leq \sqrt{n}$, and p and q are relatively prime to each other. Also, let equation of Bob's line L_B be $x = r$, where $0 \leq r \leq \sqrt{n}$. Observe that $A \cap B \neq \emptyset$ if and only if L_A and L_B intersects at a point of $G_{(0,0)}$. Moreover, L_A and L_B intersects at a grid point if and only if q divides r and $1 \leq \frac{pr}{q} \leq \sqrt{n}$. So, Alice and Bob run the communication protocol for $\textsc{Gcd}_{\sqrt{n}}(q, r)$ to decide whether $q = \gcd(q, r)$. If $q = \gcd(q, r)$ and $1 \leq \frac{pr}{q} \leq \sqrt{n}$ (again Alice and Bob can decide this using $O(1)$ bits of communications) then $A \cap B \neq \emptyset$, otherwise $A \cap B = \emptyset$. Alice and Bob can decide if $q = \gcd(q, r)$ and $1 \leq \frac{pr}{q} \leq \sqrt{n}$ using $O(1)$ bits of communication.

[8] With out loss of generality assume that $\sqrt{n}$ is an integer

[9] Recall that we have assumed, without loss of generality, that d divides n.

The communication cost of our protocol is dominated by the communication complexity of $\textsc{Gcd}_{\sqrt{n}}(q, r)$, which is equal to $O\left(\frac{\log n}{\log\log n} \log\log\log n \log\frac{1}{\delta}\right)$ by Observation 23. ◀

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 Joshua Brody, Amit Chakrabarti, Ranganath Kondapally, David P. Woodruff, and Grigory Yaroslavtsev. Beyond Set Disjointness: The Communication Complexity of Finding the Intersection. In *ACM Symposium on Principles of Distributed Computing, PODC*, pages 106–113, 2014.

3 Bernard Chazelle. *The Discrepancy Method: Randomness and Complexity*. Cambridge University Press, 2001.

4 Anirban Dasgupta, Ravi Kumar, and D. Sivakumar. Sparse and Lopsided Set Disjointness via Information Theory. In *Proceedings of the RANDOM-APPROX*, pages 517–528, 2012.

5 Sariel Har-Peled. *Geometric Approximation Algorithms*. Number 173 in Mathematical Surveys and Monographs. American Mathematical Soc., 2011.

6 Johan Håstad and Avi Wigderson. The Randomized Communication Complexity of Set Disjointness. *Theory of Computing*, 3(1):211–219, 2007.

7 T. S. Jayram, Ravi Kumar, and D. Sivakumar. Two Applications of Information Complexity. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, STOC*, pages 673–682, 2003.

8 Bala Kalyanasundaram and Georg Schnitger. The Probabilistic Communication Complexity of Set Intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992.

9 Eyal Kushilevitz and Noam Nisan. *Communication Complexity*. Cambridge University Press, USA, 1996.

10 Jiri Matousek. *Geometric Discrepancy: An Illustrated Guide*, volume 18. Springer Science & Business Media, 2009.

11 Jiri Matousek. *Lectures on Discrete Geometry*, volume 212. Springer Science & Business Media, 2013.

12 Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On Data Structures and Asymmetric Communication Complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998.

13 Ilan Newman. Private vs. common random bits in communication complexity. *Information Processing Letters*, 39(2):67–71, 1991.

14 János Pach and Pankaj K Agarwal. *Combinatorial Geometry*, volume 37. John Wiley & Sons, 2011.

15 Anup Rao and Amir Yehudayoff. *Communication Complexity: and Applications*. Cambridge University Press, 2020.

16 Alexander A. Razborov. On the Distributional Complexity of Disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

17 Tim Roughgarden. Communication Complexity (for Algorithm Designers). *Foundations and Trends in Theoretical Computer Science*, 11(3-4):217–404, 2016.

18 Mert Saglam and Gábor Tardos. On the Communication Complexity of Sparse Set Disjointness and Exists-Equal Problems. In *Proceedings of the 54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 678–687, 2013.

19 N. Sauer. On the Density of Families of Sets. *Journal of Combinatorial Theory, Series A*, 13(1):145–147, 1972.

20 S. Shelah. A Combinatorial Problem, Stability and Order for Models and Theories in Infinitary Languages. *Pacific Journal of Mathematics*, 41:247–261, 1972.

21 V. N. Vapnik and A. Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability and its Applications*, 16(2):264–280, 1971.

22 Andrew Chi-Chih Yao. Some Complexity Questions Related to Distributive Computing (Preliminary Report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, STOC*, pages 209–213, 1979.

A VC dimension, and Problems 1 and 2

VC dimension, and collection of d lines

Let $G \subset \mathbb{Z}^2$ be a set of n points in $\mathbb{Z}^2$. Observe, that the communication functions $\textsc{Disj}_G\,|_{\mathcal{L}\times\mathcal{L}}$ (defined in Problem 1) and $\textsc{Disj}_G\,|_{\mathcal{G}\times\mathcal{G}}$, where

$$\mathcal{G} = \left\{ G \cap \left(\bigcup_{1\le j\le d} \ell_j \right) \mid \{\ell_1, \ldots, \ell_d\} \in \mathcal{L} \right\},$$

are equivalent problems. Note that the set $\mathcal{L}$ is defined in Problem 1. Using standard geometric arguments, see [11, Chap. 10] and [5, Chap. 5], we can show that VC-dim$(\mathcal{G}) = 2d$.

VC dimension, and collection of d intervals

Let $X \subset \mathbb{Z}$ be a set of n points in $\mathbb{Z}$. Observe, that the communication functions $\textsc{Disj}_X\,|_{\mathcal{I}\times\mathcal{I}}$ (defined in Problem 2) and $\textsc{Disj}_X\,|_{\mathcal{F}\times\mathcal{F}}$, where

$$\mathcal{F} = \left\{ X \cap \left(\bigcup_{1\le j\le d} I_j \right) \mid \{I_1, \ldots, I_d\} \in \mathcal{I} \right\},$$

are equivalent problems. Note that the set $\mathcal{I}$ is defined in Problem 2. Using standard geometric arguments, as in the above case, we can show that VC-dim$(\mathcal{F}) = 2d$.

Testing Data Binnings

Clément L. Canonne
IBM Research, Almaden, CA, USA
ccanonne@cs.columbia.edu

Karl Wimmer
Duquesne University, Pittsburgh, PA, USA
wimmerk@duq.edu

Abstract

Motivated by the question of data quantization and "binning," we revisit the problem of identity testing of discrete probability distributions. Identity testing (a.k.a. one-sample testing), a fundamental and by now well-understood problem in distribution testing, asks, given a reference distribution (model) $\mathbf{q}$ and samples from an unknown distribution $\mathbf{p}$, both over $[n] = \{1, 2, \ldots, n\}$, whether $\mathbf{p}$ equals $\mathbf{q}$, or is significantly different from it.

In this paper, we introduce the related question of *identity up to binning*, where the reference distribution $\mathbf{q}$ is over $k \ll n$ elements: the question is then whether there exists a suitable binning of the domain $[n]$ into k intervals such that, once "binned," $\mathbf{p}$ is equal to $\mathbf{q}$. We provide nearly tight upper and lower bounds on the sample complexity of this new question, showing both a quantitative and qualitative difference with the vanilla identity testing one, and answering an open question of Canonne [6]. Finally, we discuss several extensions and related research directions.

2012 ACM Subject Classification Theory of computation → Machine learning theory; Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases property testing, distribution testing, identity testing, hypothesis testing

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.24

Category RANDOM

Related Version A full version of this paper is available at `https://arxiv.org/abs/2004.12893`.

1 Introduction

Distribution testing [3], an area at the intersection of computer science, data science, and statistics which emerged as an offspring of the field of property testing [19, 15], concerns itself with the following type of questions: "upon observing independent data points originating from some unknown process or probability distribution $\mathbf{p}$, can we quickly and efficiently decide whether $\mathbf{p}$ satisfies some desirable property, or significantly violates this property?" One of the prototypical instances of this is the question of *identity testing* (also commonly known as one-sample testing, or goodness-of-fit), where one is given a reference distribution $\mathbf{q}$ and aims to test whether the unknown $\mathbf{p}$ is equal to this purported model $\mathbf{q}$, or far from it in statistical distance.

The sample and time complexity of identity testing have been thoroughly studied, and this question is now well-understood with regard to all parameters at play (see [16, 2, 18, 13, 11, 14, 10, 21, 4], or the surveys [5, 1]). However, at the very heart of the question's formulation lies a significant assumption: namely, that the *domain* of the distributions, generally taken to be the set $[n] \coloneqq \{1, 2, \ldots, n\}$, is the "right" representation of the data. In many situations, this is not the case: the observations are made with a given (often arbitrary) level of granularity, e.g., imposed by the accuracy of the measuring equipment; this may lead to falsely (over)accurate measurements, with non-significant precision in the observations. In such cases, the domain $[n]$ of both the model $\mathbf{q}$ and the distribution of the measurements $\mathbf{p}$


© Clément L. Canonne and Karl Wimmer;
licensed under Creative Commons License CC-BY



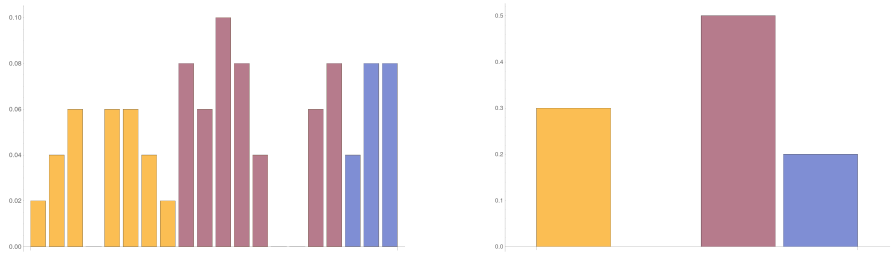
are somewhat of a red herring, and relying on it to perform identity testing may lead to a wrong answer, by introducing discrepancies where there is none. Instead, a more robust approach is to decide if $\mathbf{p}$ conforms to $\mathbf{q}$ on some suitable quantization of the data: which leads to the question, first suggested as an open question in [6] and introduced in this work, of *testing identity up to binning*.

In more detail, the question we consider is the following:

> **Identity-Up-To-Binning**$(n, \mathbf{q}, k, \varepsilon)$: Given a target number of bins k and a reference distribution $\mathbf{q}$ over $[k]$, a distance parameter ε, and i.i.d. samples from an unknown distribution $\mathbf{p}$ over $[n]$, is there a partition[1] of the domain in k intervals $I_1, \ldots, I_k$ such that $\mathbf{p}(I_j) = \mathbf{q}(j)$ for all $1 \leq j \leq k$, or is $\mathbf{p}$ at total variation distance at least ε from all distributions for which such a binning exists?

Before proceeding further, let us discuss some aspects of this formulation (which is illustrated in Figure 1). First of all, this formulation is intrinsically tied to discrete distributions (which aligns with the motivations laid out earlier, relating to discretization of observations): indeed, if $\mathbf{p}$ is continuous, then for all choices of $\mathbf{q}$ there always exists a suitable partition of the domain; so that $\mathbf{p}$ trivially satisfies the property. This formulation also allows us to test for some underlying structure, without caring about other irrelevant details of the distribution. For instance, setting $\mathbf{q} = (1/3, 1/3, 1/3)$ captures all distributions $\mathbf{p}$ with three disjoint groups of elements, separated by an arbitrary number of zero-probability elements, each with total probability mass $1/3$.

In terms of the range of parameters, we encourage the reader to keep in mind the setting where $k \ll n$ (or even where k is a relatively small constant), which captures fitting an over-accurate set of measurements to a simple model. Yet, we emphasize that even the case $k = n$ is of interest and does not collapse to identity testing. Indeed, due to our allowing empty intervals, the problem is both qualitatively and quantitatively different from identity testing, and can be interpreted as identity testing up to merging some clusters of adjacent domain elements and having zero-probability elements in the reference distributions for data points considered irrelevant (see Figure 2 for a simple illustration).



(a) Distribution $\mathbf{p}$ on $[n]$. **(b)** Distribution $\mathbf{q}$ on $[k]$.

Figure 1 An example, for $n = 20$ and $k = 4$. Here, $\mathbf{q} = (3/10, 0, 1/2, 1/5)$, and $\mathbf{p} = \frac{1}{50}(1, 2, 3, 0, 3, 3, 2, 1, 4, 3, 5, 4, 2, 0, 0, 3, 4, 2, 4, 4)$. A possible partitioning is $I_1 = \{1, 2, \ldots, 8\}$, $I_2 = \emptyset$, $I_3 = \{9, \ldots, 17\}$, and $I_4 = \{18, 19, 20\}$.

[1] Throughout this paper, by *partitions* we refer to ordered partitions with possibly empty subsets: i.e., a partition of $[n]$ in k sets is a sequence of pairwise disjoint sets $(I_1, \ldots, I_k)$ such that $\cup_{j=1}^{k} I_j = [n]$, with $|I_j| \geq 0$ for all j.

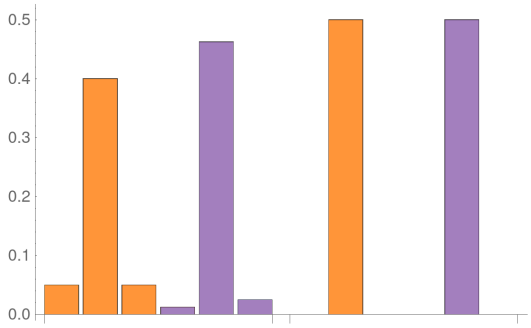


Figure 2 An example for $n = k = 6$. Here, $\mathbf{q} = (0, 1/2, 0, 0, 1/2, 0)$ (on the right), and $\mathbf{p} = (1/20, 2/5, 1/20, 1/80, 37/80, 1/40)$ (on the left). The reference distribution $\mathbf{q}$ focuses on the two modes of the distribution (writing off the other elements as "noise" or spurious), while the sampled distribution $\mathbf{p}$ also features those elements.

We further note that one could consider variants of this problem, each with a slightly different focus. The first variant, *simultaneous binning*, would feature a reference distribution $\mathbf{q}$ over $[n]$ (instead of a set of bin probabilities $\mathbf{q}$ on $[k]$), and ask about the existence of an interval partition $I_1, \ldots, I_k$ such that $\mathbf{p}(I_j) = \mathbf{q}(I_j)$ for all j. One interesting aspect of this variant is that it generalizes non-trivially to continuous distributions (or mixtures of continuous and discrete). However, even insisting for non-empty intervals I_j's, this formulation is only interesting for distributions putting significant probability mass on the first $k-1$ elements.[2] For $k = o(n)$, and in particular constant n, this is a significant restriction. A second variant, which suffers the same drawback but allows more flexibility, also provides a reference distribution $\mathbf{q}$ over $[n]$, but only asks about the existence of *independent binnings*: two interval partitions $I_1, \ldots, I_k$ and $I'_1, \ldots, I'_k$ such that $\mathbf{p}(I_j) = \mathbf{q}(I'_j)$ for all j.

We note that our upper bound (Theorem 1) easily extends to these two different variants as well; and our lower bound (Theorem 2) applies to the last independent binnings variant as well.

1.1 Our results and techniques

In this paper, we establish nearly matching upper and lower bounds on the **Identity-Up-To-Binning** problem. Our first result is an efficient algorithm for testing the property:

▶ **Theorem 1.** *For every domain size n, number of bins $1 \leq k \leq n$, and fixed reference distribution $\mathbf{q}$ over $[k]$, there exists a (computationally efficient) algorithm for testing identity-up-to-binning to $\mathbf{q}$ with sample complexity $O\big(k/\varepsilon^2\big)$, where $\varepsilon \in (0, 1]$ is the distance parameter.*

Crucially, this upper bound is *independent* of the underlying domain size n of the unknown distribution, and only depends, linearly, on the target number of bins k. This is to be compared to the "standard" identity testing problem, which is known to have sample complexity $\Theta(\sqrt{n})$. One may wonder if the dependence on the number of bins in the above theorem is necessary, or if one could hope to achieve a *sublinear* dependence on k. Our next (and main) result shows that a near-linear dependence is unavoidable, ruling out any strongly sublinear dependence on k:

[2] Indeed, unless both the reference $\mathbf{q}$ and the unknown $\mathbf{p}$ put $\Omega(\varepsilon)$ mass on $\{1, 2, \ldots, k-1\}$, a trivial partitioning with $I_j = \{j\}$ for $1 \leq j \leq k-1$ and $I_k = \{k, \ldots, n\}$ provides an immediate answer to the problem.

▶ **Theorem 2** (Informal version of Theorem 6). *For every $k \geq 1$, there exists $n = n(k)$ and an absolute constant $\varepsilon_0 > 0$ such that the following holds. There exists a reference distribution $\mathbf{q}$ over $[k]$ such that any algorithm for testing identity-up-to-binning to $\mathbf{q}$ to distance ε_0 must have sample complexity $\Omega\big(k^{1-o(1)}\big)$.*

Note that by standard techniques, this readily implies a lower bound of $\Omega\big(k^{1-o(1)}/\varepsilon\big)$, for all $\varepsilon \in (0, \varepsilon_0]$. We further conjecture the tight bound to be $\Omega\big(k/\varepsilon^2\big)$ (i.e., matching our upper bound). It is worth noting that the obvious and natural approach, a reduction from (standard) identity testing, would fall short of this goal, as it would only lead to a much weaker $\Omega(\sqrt{k}/\varepsilon^2)$ lower bound.

Our techniques

Our upper bound proceeds via the "testing-by-learning" paradigm. Specifically, given a reference distribution $\mathbf{q}$ over $[k]$, and sample access to an unknown distribution $\mathbf{p}$ over $[n]$, we learn a hypothesis distribution $\hat{\mathbf{p}}$ such that, informally, for every binning of $[n]$ into k intervals, the distributions $\mathbf{p}$ and $\hat{\mathbf{p}}$ are close. (More formally, the distribution $\hat{\mathbf{p}}$ is close to $\mathbf{p}$ in $\mathcal{A}_k$-distance, which we define later.) Since we have exact access to $\hat{\mathbf{p}}$, we can enumeratively test every potential binning. Although this does not save on the query complexity, we are able to use dynamic programming to find the best binning in time polynomial in n and k.

The proof of our lower bound is significantly more involved and proceeds in three stages, which we outline here. We first show that, if n is sufficiently large as a function of k, then we may assume that the algorithm only looks at the ordering of the samples received from $[n]$, instead of at their actual "names" (we refer to the ordering as the *ordered fingerprint* of the sample). Our bound uses a Ramsey theory argument very similar to that of [12].

We then show, for every integer m, the existence of two different distributions $\mathbf{p}'$ and $\mathbf{q}'$ over $[\mathrm{poly}(m) \cdot 2^m]$ such that (i) $\mathbf{p}'$ and $\mathbf{q}'$ are far in total variation distance, and (ii) given the ordered fingerprint of m samples generated by one of $\mathbf{p}'$ and $\mathbf{q}'$, it is information-theoretically impossible to determine which of $\mathbf{p}'$ and $\mathbf{q}'$ generated the ordered fingerprint. There are 2^m such ordered fingerprints, one for every composition of m, so it is unclear if the support size of the distributions in this claim can be significantly reduced. While this is an independently interesting problem, any progress on this problem would only directly result in a decrease of the $o(1)$ exponent in our main theorem.

Finally, using the above pair of distributions as "buckets" of two distributions $\mathbf{p}$ and $\mathbf{q}$, respectively, we show that any algorithm distinguishes $\mathbf{p}$ and $\mathbf{q}$ and only considers ordered fingerprints must have access to at least $m + 1$ samples from at least one of the (assuming m is independent of k) $\Omega(k)$ buckets. By collision bounds for the generalized birthday problem (see also [20] for more formal statements), with high probability we need a sample size of $\Omega(k^{1-1/(m+1)})$ to ensure that the algorithm succeeds.

1.2 Future directions

We conclude with a few research directions we deem particularly promising. A first natural question is to understand the analogue of the **Identity-Up-To-Binning** problem for *closeness* testing, that is, when both $\mathbf{p}$ (over $[n]$) and $\mathbf{q}$ (over $[k]$) are unknown and available only through i.i.d. samples. This would in particular capture situations where data is collected from two different sources (or sensors), each using a different discretization scheme or with different precision, and one aims at deciding whether the underlying distributon is the same.

Another avenue is to consider the *tolerant* testing version of the problem, that is, to decide whether $\mathbf{p}$ is *close* to a binning of $\mathbf{q}$ (versus far from any such binning). It is known that tolerant identity testing has significantly larger (namely, nearly linear in the domain size) sample complexity than identity testing; whether this is still the case for their identity-up-to-binning analogues is an intriguing question.

Lastly, it would be interesting to generalize the question to other partially-ordered domains (e.g., the hypergrid $[n]^d$, or the Boolean hypercube $\{0,1\}^n$), for the suitable notion of "interval" in these posets.

2 Preliminaries

All throughout the paper, we write log for the binary logarithm. Given integers n and k, we write $\mathcal{J}_{n,k}$ for the set of all $\binom{n+k-1}{n}$ partitions of $[n]$ in k consecutive (and possibly empty) pairwise disjoint intervals. Recall that the total variation distance between two distributions $\mathbf{p}, \mathbf{p}'$ over $[n]$ is $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}_1, \mathbf{p}_2) = \sup_{S \subseteq [n]} (\mathbf{p}(S) - \mathbf{p}'(S))$, and is equal to $\frac{1}{2}\sum_{i=1}^{n} |\mathbf{p}(i) - \mathbf{p}'(i)|$, half the ℓ_1 distance between their probability mass functions.

The **Identity-Up-To-Binning**$(n, \mathbf{q}, k, \varepsilon)$ problem we consider in this paper is then formally defined as follows. Given a reference distribution $\mathbf{q}$ on $[k]$, we consider the property

$$\mathcal{P}_{\mathbf{q}} \coloneqq \left\{ \mathbf{p}' \text{ distribution on } [n] \,:\, \exists (I_1, \dots, I_k) \in \mathcal{J}_{n,k}\,,\ \max_{j \in [k]} |\mathbf{p}'(I_j) - \mathbf{q}(j)| = 0 \right\} \tag{1}$$

The question is then, given a distance parameter $\varepsilon \in (0,1]$, to distinguish (i) $\mathbf{p} \in \mathcal{P}_{\mathbf{q}}$ from (ii) $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}}) > \varepsilon$, where $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}}) = \min_{\mathbf{p}' \in \mathcal{P}_{\mathbf{q}}} \mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathbf{p}')$. We will rely extensively, for our lower bounds, on the below fact, which allows us to lower bound $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}})$ by a more actionable quantity.

▷ Claim 3. For any distributions $\mathbf{p}$ and $\mathbf{q}$ (over $[n]$ and $[k]$, respectively) we have

$$\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}}) \geq \frac{1}{2}\,\mathrm{dist}(\mathbf{p}, \mathbf{q}) \coloneqq \min_{f} \sum_{j=1}^{k} \Big| \mathbf{q}(j) - \sum_{t : f(t) = j} \mathbf{p}(t) \Big|,$$

where the minimum is taken over all nondecreasing functions $f \colon [n] \to [k]$.

Proof. Fix $\mathbf{p}, \mathbf{q}$ as in the statement. One can think of such a nondecreasing f as defining a decomposition of $[n]$ into k disjoint (and possible empty) intervals $I_1, \dots, I_k$, where $I_j = f^{-1}(j)$. In particular, let $\mathbf{p}^* \in \mathcal{P}_{\mathbf{q}}$ such that $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathbf{p}^*) = \mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}})$, and $I_1^*, \dots, I_k^*$ the corresponding partition for $\mathbf{p}^*$. Define $f^* \colon [n] \to [k]$ from this partition by setting $f(t) \coloneqq \sum_{j=1}^{k} j \mathbb{1}_{I_j^*}(t)$: f^* is then non-decreasing, and

$$\sum_{j=1}^{k} \Big| \mathbf{q}(j) - \sum_{t : f^*(t) = j} \mathbf{p}(t) \Big| = \sum_{j=1}^{k} \big| \mathbf{q}(j) - \mathbf{p}(I_j^*) \big| = \sum_{j=1}^{k} \big| \mathbf{p}^*(I_j^*) - \mathbf{p}(I_j^*) \big| \leq \sum_{i=1}^{n} \big| \mathbf{p}^*(i) - \mathbf{p}(i) \big|$$

where the last equality is due to the fact that $\mathbf{p}^*(I_j^*) = \mathbf{q}(j)$ for all $j \in [k]$ (since $\mathbf{p}^* \in \mathcal{P}_{\mathbf{q}}$), and last inequality is the triangle inequality. Since the RHS equals $2\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathbf{p}^*)$, we get $\min_f \sum_{j=1}^{k} |\mathbf{q}(j) - \sum_{t : f(t) = j} \mathbf{p}(t)| \leq 2\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}})$. ◁

3 Algorithms

In this section, we prove Theorem 1, restated below:

▶ **Theorem 4.** *For any fixed reference distribution* $\mathbf{q}$ *over* $[k]$ *and number of bins* $1 \leq k \leq n$*, there exists a (computationally efficient) algorithm for* **Identity-Up-To-Binning**$(n, \mathbf{q}, k, \varepsilon)$ *with sample complexity* $O\big(k/\varepsilon^2\big)$*.*

Proof. We will use the following notion of $\mathcal{A}_\ell$*-distance*, which interpolates between Kolmogorov distance ($\ell = 2$) and (twice the) total variation ($\ell = n$) [9, 7, 13, 12]: for any two distributions $\mathbf{p}', \mathbf{q}'$ over $[n]$,

$$\|\mathbf{p}' - \mathbf{q}'\|_{\mathcal{A}_\ell} \coloneqq \max_{(I_1,\dots,I_\ell)\in\mathcal{J}_{n,\ell}} \sum_{j=1}^{\ell} |\mathbf{p}'(I_j) - \mathbf{q}'(I_j)| \tag{2}$$

One can check that this defines a *bona fide* norm. Further, it is known, from the VC inequality, that $O\big(\ell/\varepsilon^2\big)$ samples are sufficient to learn any distribution in $\mathcal{A}_\ell$-distance ε (and with failure probability 1/6) [9, 7], and, further, that the empirical estimator achieves this bound. Using this, the algorithm is as follows:

1. Learn the unknown $\mathbf{p}$ to $\mathcal{A}_k$-distance ε, with failure probability 1/6. Call the result (the empirical estimator, which is a distribution over $[n]$) $\hat{\mathbf{p}}$.
2. Compute the minimum $\Delta_{\mathbf{q}}(\mathcal{I}, \hat{\mathbf{p}}) \coloneqq \sum_{j=1}^{k} |\hat{\mathbf{p}}(I_j) - \mathbf{q}(j)|$ over all partitions $(I_1, \dots, I_k) \in \mathcal{J}_{n,k}$ in intervals such that $|I_j| > 0$ whenever $\mathbf{q}(j) > 0$.
3. If there exists $\mathcal{I}$ such that $\Delta_{\mathbf{q}}(\mathcal{I}, \hat{\mathbf{p}}) \leq \varepsilon$, return accept; otherwise, return reject.

Note that only the first step requires samples from $\mathbf{p}$, so that the sample complexity is indeed $O\big(k/\varepsilon^2\big)$. The second step is purely computational, and can be implemented in time $\mathrm{poly}(n, k)$ via a simple dynamic programming approach.

We now argue correctness. With probability at least 5/6, the first step produced a correct $\hat{\mathbf{p}}$, i.e., one that is ε-close to $\mathbf{p}$ in $\mathcal{A}_k$-distance; we hereafter assume this holds.

Completeness. Suppose $\mathbf{p} \in \mathcal{P}_{\mathbf{q}}$, and let $\mathcal{I}^* = (I_1^*, \dots, I_k^*) \in \mathcal{J}_{n,k}$ be any partition witnessing it. Note that this partition then satisfies $|I_j^*| > 0$ for all j such that $\mathbf{q}(j) > 0$, as otherwise the distance after binning is positive. Then,

$$\Delta_{\mathbf{q}}(\mathcal{I}^*, \hat{\mathbf{p}}) = \sum_{j=1}^{k} \big|\hat{\mathbf{p}}(I_j^*) - \mathbf{q}(j)\big| \leq \sum_{j=1}^{k} \big|\hat{\mathbf{p}}(I_j^*) - \mathbf{p}(I_j^*)\big| + \sum_{j=1}^{k} \big|\mathbf{p}(I_j^*) - \mathbf{q}(j)\big| \leq \|\hat{\mathbf{p}} - \mathbf{p}\|_{\mathcal{A}_k} \leq \varepsilon$$

the second-to-last inequality by definition of $\mathcal{A}_k$-distance and the fact that $\sum_{j=1}^{k} |\mathbf{p}(I_j^*) - \mathbf{q}(j)| = 0$. Therefore, the algorithm will find a good partition and output accept.

Soundness. Suppose now by contrapositive that the algorithm outputs accept. This means it found, in step 2, some partition $\mathcal{I}^* = (I_1^*, \dots, I_k^*) \in \mathcal{J}_{n,k}$ such that $\Delta_{\mathbf{q}}(\mathcal{I}^*, \hat{\mathbf{p}}) \leq \varepsilon/4$. But then,

$$\sum_{j=1}^{k} \big|\mathbf{p}(I_j^*) - \mathbf{q}(j)\big| \leq \sum_{j=1}^{k} \big|\mathbf{p}(I_j^*) - \hat{\mathbf{p}}(I_j^*)\big| + \sum_{j=1}^{k} \big|\hat{\mathbf{p}}(I_j^*) - \mathbf{q}(j)\big| \leq \|\hat{\mathbf{p}} - \mathbf{p}\|_{\mathcal{A}_k} + \Delta_{\mathbf{q}}(\mathcal{I}^*, \hat{\mathbf{p}}) \leq 2\varepsilon$$

the second-to-last inequality again by definition of $\mathcal{A}_k$-distance. We claim that this implies that $\mathbf{p}$ is ε-close (in total variation distance) to some $\mathbf{p}^* \in \mathcal{P}_{\mathbf{q}}$. Indeed, we can build $\mathbf{p}^*$ in a greedy fashion from $\mathbf{p}$: as long as there exist $I_{j_1}^*$, $I_{j_2}^*$ such that $\mathbf{p}(I_{j_1}^*) > \mathbf{q}(j_1)$ but $\mathbf{p}(I_{j_2}^*) < \mathbf{q}(j_2)$, we move $\delta \coloneqq \min(|\mathbf{p}(I_{j_1}^*) - \mathbf{q}(j_1)|, |\mathbf{p}(I_{j_2}^*) - \mathbf{q}(j_2)|)$ probability mass from

(arbitrary) elements of $I^*_{j_1}$ to an arbitrary element of $I^*_{j_2}$. Here, we used our condition on the partition found in step 2, which ensures $I^*_{j_2}$ is non-empty. This incurs total variation δ (from $\mathbf{p}$), and reduces $\sum_{j=1}^{k} |\mathbf{p}(I^*_j) - \mathbf{q}(j)|$ by 2δ. Repeating until it is no longer possible, we obtain the claimed $\mathbf{p}^*$, and therefore that $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}}) \leq \varepsilon$. ◀

▶ Remark 5. As mentioned in the introduction, it is straightforward to adapt the algorithm and argument to either the "simultaneous binning" and "independent binnings" variants of the problem, leading to the same sample complexity upper bound (and time complexity) for these related questions.

4 Lower Bounds

In this section, we prove our main theorem (Theorem 2), restated below:

▶ **Theorem 6.** *For every $k \geq 1$, there exists $n = n(k)$ such that the following holds. There exists a reference distribution $\mathbf{q}$ over $[k]$ such that any algorithm for* **Identity-Up-To-Binning**$(n, \mathbf{q}, k, \varepsilon)$ *must have sample complexity $\frac{k}{2^{O(\sqrt{\log k})}}$.*

At a high level, the proof of Theorem 6 consists of three parts:

(1) First, we show that, for every n, there is an $N(n)$ such that, if there is a algorithm that succeeds in the data binning problem over $[N(n)]$ with s samples, then there is an algorithm that succeeds in the data binning problem over $[n]$ with s samples, such that this latter algorithm only considers the "ordered fingerprint" of the sample. This approach, analogous to that of [12], uses Ramsey theory, and enables us to restrict ourselves to (simpler to analyze) order-based algorithms. (Proposition 9)

(2) Second, we show for every constant m, there exist $b(m)$ and two distributions $\mathbf{p}'$ and $\mathbf{q}'$ on $[b(m)]$ that (i) are "far from being cyclic shifts" of one another (which will allow us to argue about distance to being a binning on k intervals later), and (ii) cannot be distinguished from ordered fingerprints from s samples. The function $b(m)$ is exponential in m. (Lemma 12)

(3) Third, we consider the distributions $\mathbf{p}$ and $\mathbf{q}$ on $[k]$ constructed such that the distribution restricted to each block of $b(m)$ elements is $\mathbf{p}'$ and $\mathbf{q}'$, respectively, in the natural way. We show that (i) $\mathbf{p}$ is "far" from any binning of $\mathbf{q}$, and (ii) distinguishing these two distributions with ordered fingerprints requires seeing $m+1$ samples from one particular block. By folklore collision bounds, this implies a $\Omega(k^{1-1/(m+1)})$ sample lower bound for distinguishing $\mathbf{p}$ and $\mathbf{q}$ from ordered fingerprints. (Corollary 16)

Combining the three (applying 1 with n set to k, and m in 3 set to $\sqrt{\log k}$) then yields the theorem.

4.1 From samples to ordered fingerprints

The first step of our reduction will consist in "hiding" information from the algorithms. Roughly speaking, for the sake of our lower bound analysis we would like to argue that, without loss of generality, any testing algorithm can be assumed to only be given the *order relations* between the samples from the unknown distribution, instead of the samples themselves. For instance, instead of seeing four samples $12, 7, 98, 7$, we wish to restrict our analysis to algorithms which only see that $x_{(1)} = x_{(2)} < x_{(3)} < x_{(4)}$. Since we are drawing i.i.d. samples from the distribution, the order that the samples come in does not matter.

However, this simplifying assumption is not actually without loss of generality, and does not quite hold as stated: an algorithm *can* sometimes infer more information from the values of the samples from $\mathbf{p}$ than from their ordered relations alone. We will prove a weaker statement, sufficient for our purposes; in order to do so, we start by introducing some notions formalizing the aforementioned "order relations."

▶ **Definition 7** (Ordered fingerprints). *The* ordered fingerprint *of a sequence of s values in $[n]$ is an ordered frequency vector of the elements occurring in the sequence, with labels removed. Formally, for each element $i \in [n]$, let $\boldsymbol{F}_i$ be the number of times element i appears, and let $j_1 < j_2 < \ldots < j_t$ be the indices such that $\boldsymbol{F}_{j_i}$ is positive (so that t is the number of distinct elements in the sequence). The ordered fingerprint is then the ordered collection of positive integers $(\boldsymbol{F}_{j_1}, \boldsymbol{F}_{j_2}, \ldots, \boldsymbol{F}_{j_t})$ such that $\sum_{i=1}^{t} \boldsymbol{F}_{j_i} = s$.*

To later argue about indistinguishability of our instances, we will also rely on the below notion of s-way moments (induced by an ordered fingerprint):

▶ **Definition 8** (s-way moments). *Given a distribution $\mathbf{p}$ over $[n]$ and an ordered tuple of positive integers $F = (F_1, F_2, \ldots, F_t)$ such that $\sum_{i=1}^{t} F_i = s$, the probability that the ordered fingerprint on s samples is $(F_1, F_2, \ldots, F_t)$ is given by*

$$\mathbf{p}^F \coloneqq \binom{s}{F_1, F_2, \ldots, F_t} \sum_{1 \le i_1 < i_2 < \cdots < i_t \le n} \prod_{j=1}^{t} \mathbf{p}(i_j)^{F_j} \tag{3}$$

We call such an expression an s-way moment of $\mathbf{p}$. *Note that the s-way moments of $\mathbf{p}$ completely determine the distribution of the ordered fingerprint of s random samples drawn from $\mathbf{p}$. Thus, for two distributions $\mathbf{p}$ and $\mathbf{q}$, if $\mathbf{p}^F = \mathbf{q}^F$ for all F such that $|F| = s$, then $\mathbf{p}$ and $\mathbf{q}$ cannot be distinguished from ordered fingerprints on s samples.*[3]

We now state and prove the following lemma, which captures the intuition discussed above and will be the first component of our lower bound:

▶ **Proposition 9.** *For every $n \ge 1$ and $s \ge 1$, there exists $N \ge 1$ such that the following holds for every $k \ge 1$, reference distribution $\mathbf{q}$ over $[k]$, and $\varepsilon \in (0, 1]$. If there exists an algorithm $\mathcal{A}_N$ that, for every distribution $\mathbf{p}'$ over $[N]$, solves the problem* **Identity-Up-To-Binning**$(N, \mathbf{q}, k, \varepsilon)$ *with s samples from $\mathbf{p}'$, then there exists an algorithm $\mathcal{A}_n$ that, for every distribution $\mathbf{p}$ over $[n]$, solves the problem* **Identity-Up-To-Binning**$(n, \mathbf{q}, k, \varepsilon)$ *with s samples from $\mathbf{p}$. Moreover, $\mathcal{A}_n$ only considers the ordered fingerprint of the s samples.*

Proof. The argument is similar to that of Diakonikolas, Kane, and Nikishkin [12], and relies on a result of Conlon, Fox, and Sudakov:

▶ **Lemma 10** ([8]). *Given a set S and an integer t, let $\binom{S}{t}$ denote the set of subsets of cardinality t. For all positive integers, a, b, and c, there exists a positive integer N so that for any function $\chi\colon \binom{[N]}{a} \to [b]$, there exists $S \subseteq [N]$ with $|S| = c$ such that χ is constant on $\binom{S}{a}$.*

As in the proof of [12, Theorem 13], given n, s, we will invoke Lemma 10 to obtain a new domain size N, in a way detailed below: loosely speaking, a corresponds to the number of samples s, b to the number of possible decision (binary) functions from a given set of s

[3] Indeed, for every s, the $(s-1)$-way moments are linear combinations of s-way moments; we omit the details.

samples which only depend on the ordered fingerprint of those samples, and c to the target domain size s. We will use this to, given $\mathcal{A}_N$ (which induces a mapping χ from sets of s samples to such order-based decision functions), find a set $S \subseteq [N]$ of size n, and a monotone function $f\colon [n] \to [N]$ such that $f([n]) = S$. The algorithm $\mathcal{A}_s$ then runs the promised algorithm $\mathcal{A}_N$ on the distribution $f(\mathbf{p})$ obtained by applying f to the s samples. Since f is order-preserving, the distances will not change. Since we can assume that $\mathcal{A}_N$ is deterministic once we know which distinct samples we get from $[n]$, the fact that the induced χ is constant on $\binom{f([n])}{s}$ guarantees the output will be a function of the ordered fingerprint only.

However, we have an issue that [12] does not. When we draw a total of s samples, we likely will not get s distinct samples. [12] gets around this issue by "dividing" all elements of $[n]$ into sub-elements, and upon seeing an element in the sample, assign it to a uniformly random one of its sub-elements. However, our main parameter of interest k differs from theirs: if we applied this procedure to our problem, the bounds obtained would deteriorate. That is, while the $\mathcal{A}_k$-distance (the focus of [12]) would not change, the coarsening distance we consider here would.

We handle this issue as follows. Given an integer s, we will color all nonempty subsets of $[N]$ with *at most* s elements. This color associates to a set of s samples the function that $\mathcal{A}_N$ uses to accept or reject given the ordered fingerprint that accompanies these samples. Now, we apply Lemma 10 repeatedly. In our first application, we find a subset $S_s \subseteq [N]$ such that, conditioning on distinct samples, $\mathcal{A}_N$ has consistent behavior.[4] Now we remove all subsets that contain elements outside of S_s, and we apply Lemma 10 again to find a set $S_{s-1} \subseteq S_s$ such that $\mathcal{A}_N$'s behavior only depends on the ordered fingerprint given that in the s samples, there are at least $s-1$ distinct values seen. Continuing in this fashion, after applying Lemma 10 a total of s times, we arrive at a set S_1 (with $S_1 \subseteq S_2 \subseteq \cdots \subseteq S_s$) such that, conditioned on samples coming only from S_1, $\mathcal{A}_N$ depends only on the ordered fingerprint of the samples seen. In every application of the lemma, we have $a \leq s$, and since there are 2^s possible ordered fingerprints given s samples, we have $b \leq 2^{2^s}$. We set $c = n$ in our final application. ◀

4.2 Sequences far from being shifts of each other

As a building block from our lower bound construction, we will require the existence of (non-negative) sequences which are "far" from each other, in that any circular shift of the first remains far, in Hamming distance, from the second.

▶ **Definition 11** (Partial shifts). *Given an alphabet Σ and two strings x and y in Σ^n, we say that y is an r-*partial cyclic shift *of x if there exist an integer ℓ and a nondecreasing function $f\colon [n] \to [n]$ such that, for at least r indices i of $[n]$, $x_i = y_{(f(i)+\ell) \bmod n}$. That is, thinking of the symbols of y placed in a circle, y contains a substring of length r of x (without wrapping around more than once).*

For two distributions $\mathbf{p}$ and $\mathbf{q}$ over $[n]$, we then say that $\mathbf{q}$ is an *r-partial cyclic shift of $\mathbf{p}$ if the string $\mathbf{q}(1)\mathbf{q}(2)\cdots\mathbf{q}(n)$ is an r-partial cyclic shift of $\mathbf{p}(1)\mathbf{p}(2)\cdots\mathbf{p}(n)$.*

We observe that finding a witness to x and y being r-partial cyclic shifts of each other is equivalent to finding the *cyclic longest common subsequence* between x and y, which is a problem with applications in DNA sequencing [17]. With these notions in hand, we are ready to prove the key technical lemma underlying part 2 of our argument:

[4] In this case, the ordered fingerprint is the all ones vector of length s, so the functions are not very interesting.

▶ **Lemma 12.** *Let $m > 4$ be a positive integer, and $b = 5m^2 2^m$. Then there exist two distributions $\mathbf{p}'$ and $\mathbf{q}'$ over $[b]$ such that (i) $\mathbf{p}'$ and $\mathbf{q}'$ are not $(99b/100)$-partial cyclic shifts of each other, and (ii) $\mathbf{p}'$ and $\mathbf{q}'$ agree on all m-way moments.*

Proof. Let $\mathbf{p}$ be a probability distribution over $[b]$ such that its probability mass function takes the values $2/(\frac{5}{2}b)$ and $3/(\frac{5}{2}b)$ each on $b/2$ of the elements of $[b]$. Each m-way moment $\mathbf{p}^F$ can be expressed as a homogeneous polynomial in the b values $\mathbf{p}(i)$ of total degree m, such that each coefficient is 0 or $K_F := \binom{s}{F_1, F_2, \ldots, F_t}$; note that K_F is independent of $\mathbf{p}$. The number of terms of such a polynomial is at most $\binom{b+m-1}{m} \leq (b+m-1)^m \leq (2b)^m$. Thus, every m-way moment of $\mathbf{p}$ evaluates to an expression of the form $qK_F/(\frac{5}{2}b)^m$, where q is a integer such that $0 \leq q \leq (6b)^m$. Since there are 2^m m-way moments of $\mathbf{p}$, there are at most $(6b)^{m2^m}$ possible values of the m-way moments of $\mathbf{p}$.

For the claim about cyclic shifts, we use a counting argument. We will show that for any string $x \in \{2,3\}^b$, there are at most $2^{b/5}$ other strings that are $(99b/100)$-partial cyclic shifts of x. Indeed, any $(99b/100)$-partial cyclic shift y of x can be constructed by the following process: first, select $99b/100$ positions of x that will be present in y–these positions will form a witness. Next, select $99b/100$ positions of y that these bits of x will appear in, and a cyclic shift for these bits. Finally, select any values in $\{2,3\}$ for the remaining $b/100$ positions of y. Thus, the number of possible y's is at most

$$\binom{b}{b/100}^2 (99b/100) 2^{b/100} \leq 2^{2h(1/100)} 2^{b/50} \leq 2^{b/5}$$

where we have used the bound $\binom{n}{\alpha n} \leq 2^{nh(\alpha)}$, where $h(\alpha) = \alpha \log \alpha + (1-\alpha)\log(1-\alpha)$ is the binary entropy. Since there are a total of $\binom{b}{b/2} \geq 2^b/b$ strings of length b with an equal number of 2's and 3's, then for every b, we can find a set S of $2^{4b/5}/b$ strings of length b such that no two of them are $(99b/100)$-partial cyclic shifts of each other. We map strings over $\{2,3\}^b$ to distributions over $[b]$ in a natural way: given $x \in S$, we define the corresponding distribution $\mathbf{p}$ such that $\mathbf{p}(i) = x_i/(\frac{5}{2}b)$. Thus, we have $2^{4b/5}/b$ distributions over $[b]$, no two of which are $(99b/100)$-partial cyclic shifts of each other. Setting $b = 5m^2 2^m$, we then have

$$\frac{1}{b} 2^{4b/5} = \frac{2^{4m^2 2^m}}{5m^2 2^m} \geq 2^{3m^2 2^m} = (2^{3m})^{m2^m} > (30m^2 2^m)^{m2^m} = (6b)^{m2^m}.$$

Thus, by the pigeonhole principle there exist two distributions $\mathbf{p}'$ and $\mathbf{q}'$ over $[b]$ that are not $(99b/100)$-partial cyclic shifts of each other, yet such that $\mathbf{p}'$ and $\mathbf{q}'$ agree on all m-way moments. ◀

4.3 Constructing the hard-to-distinguish instances over $[n]$

For a fixed integer $m > 1$ (to be determined later) and sufficiently large k, let $\mathbf{p}'$ and $\mathbf{q}'$ be the two distributions over $[b]$ promised by Lemma 12, where $b = 5m^2 2^m$; and let $k' := k/b$. We will define $\mathbf{p}$ and $\mathbf{q}$ to be distributions over $[bk']$ in the following way: We partition $[bk']$ into blocks $B_1, B_2, \ldots, B_{k'}$, where $B_j = \{b(j-1)+1, b(j-1)+2, \ldots, bj\}$. Now $\mathbf{p}$ (resp. $\mathbf{q}$) is the distribution resulting from the following process:

- Pick a uniformly random element $i \in [k']$ and a random sample j from $\mathbf{p}'$ (resp. $\mathbf{q}'$).
- Output, as resulting sample from $\mathbf{p}$ (resp. $\mathbf{q}$), the element $(i-1)k' + j$.

Since $\mathbf{p}'(i)$ and $\mathbf{q}'(i)$ are in $\{4/(5b), 6/(5b)\}$ for $i \in [b]$, we have that $\mathbf{p}(i)$ and $\mathbf{q}(i)$ are always in $\{4/(5bk'), 6/(5bk')\}$ for all $i \in [bk']$.

We will prove that $\mathrm{d_{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}})$ is large. To facilitate this, recalling Claim 3 we will lower bound the distance of $\mathbf{p}$ to a binning of $\mathbf{q}$ in the following way:

$$\mathrm{d_{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}}) \geq \frac{1}{2}\,\mathrm{dist}(\mathbf{p}, \mathbf{q}) = \frac{1}{2}\min_{f} \sum_{i=1}^{bk'} \Big|\mathbf{q}(i) - \sum_{t:f(t)=i} \mathbf{p}(t)\Big|, \tag{4}$$

where the minimum is taken over all nondecreasing functions $f\colon [bk'] \to [bk']$. To use the language of a partition[5], the interval I_i is simply $f^{-1}(i)$. We show the following:

▷ Claim 13. There exists $\varepsilon_0 > 0$ such that, for $\mathbf{p}$ and $\mathbf{q}$ as described above, $\mathrm{d_{TV}}(\mathbf{p}, \mathcal{P}_{\mathbf{q}}) \geq \varepsilon_0$. (Moreover, one can take $\varepsilon_0 = 10^{-7}$.)

Proof. We will show that $\mathrm{dist}(\mathbf{p}, \mathbf{q}) \geq 1/(5 \cdot 10^6)$. We first rewrite the distance as $\mathrm{dist}(\mathbf{p}, \mathbf{q}) = \min_f \sum_{j=1}^{k'} d(B_j)$, where, using the same blocks as described earlier, we define

$$d(B_j) \coloneqq \sum_{i \in B_j} \Big|\mathbf{q}(i) - \sum_{t:f(t)=i} \mathbf{p}(t)\Big|$$

(suppressing its dependence on $\mathbf{p}, \mathbf{q}$, and f). We will call a block B_j *good* if $d(B_j) \leq 1/(2500k')$. Note that f need not be a function that maps elements of B_j to other elements of B_j. Since $\mathbf{p}(i)$ and $\mathbf{q}(i)$ are always in $\left\{\frac{4}{5bk'}, \frac{6}{5bk'}\right\}$, any element $i \in B_j$ such that $|f^{-1}(i)| \neq 1$ will contribute at least $\frac{2}{5bk'}$ to $d(B_j)$. Thus, we call an element i *good* if there is a unique t such that $f(t) = i$ and $\mathbf{q}(i) = \mathbf{p}(t)$. Since each element that is not good contributes $\frac{2}{5bk'}$ to $d(B_j)$, it follows that, in a good block, at most $1/(2500k')/(\frac{2}{5bk'}) = b/1000$ elements are not good, and therefore a good block must contain at least $999b/1000$ good elements.

Let B_j be a good block: it must be the case that for at least $999b/1000$ values of $g \in B_j$, there exists a unique t such that $f(t) = g$. Let $g_1 < g_2 < \ldots < g_v$ be the good elements in B_j (where $v \geq 999b/1000$), and let $t_1, t_2, \ldots, t_v$ be such that $f(t_i) = g_i$ for $i = 1, 2, \ldots, v$. By assumption, the distributions $\mathbf{p}$ and $\mathbf{q}$ are not $(99b/100)$-partial cyclic shifts of each other. Thus, for every i' and i'' where $i' < i''$, if $t_{i''} - t_{i'} < b$, then $i'' - i' \leq 99b/100$, or else the good elements of B_j and their preimages under f would be a witness to $\mathbf{p}'$ and $\mathbf{q}'$ being $(99b/100)$-partial cyclic shifts of one another. It follows that

$$t_v - t_1 \geq b + (999b/1000 - 99b/100) \geq (1 + 1/1000)b.$$

That is, every good block "uses up" a bit more than b elements of the domain of f. Since $f\colon [bk'] \to [bk']$ must be a nondecreasing function, we can have at most $\dfrac{bk'}{(1+1/1000)b} \leq (1 - 1/2000)k'$ good blocks. Thus, at least $k'/2000$ of the blocks are not good. It follows that

$$\mathrm{dist}(\mathbf{p}, \mathbf{q}) = \min_f \sum_{j=1}^{k'} d(B_j) \geq (k'/2000)(1/2500k') = 1/(5 \cdot 10^6),$$

as claimed. ◁

We are now in position to conclude the step 3 of our overall proof, establishing the following lower bound:

[5] Recall that the sets in the partition are allowed to be empty.

▶ **Theorem 14.** *There exists an absolute constant $\varepsilon_0 > 0$ such that, for every sufficiently large integers $k, m \geq 1$, there exists $\mathbf{q}$ over $[k$ for which testing* **Identity-Up-To-Binning**$(k, \mathbf{q}, k, \varepsilon_0)$ *from ordered fingerprints only requires $\Omega\big((k/m^2 2^m)^{1-1/m}\big)$ samples. (Moreover, one can take $\varepsilon_0 = 10^{-7}$.)*

Proof. Let $\mathbf{p}$ and $\mathbf{q}$ be the distributions described above, with $k = bk'$ and $\varepsilon_0 \coloneqq 10^{-7}$ as in Claim 13. We consider the problem **Identity-Up-To-Binning**$(k, \mathbf{q}, k, \varepsilon_0)$. We will show that any algorithm that takes too few samples, and only considers their ordered fingerprint, can distinguish between $\mathbf{p}$ and $\mathbf{q}$ with probability at most 1/3. Since $\mathrm{d}_{\mathrm{TV}}(\mathbf{p}, \mathbf{q}) > \varepsilon_0 \coloneqq 10^{-7}$ (and, of course, $\mathrm{d}_{\mathrm{TV}}(\mathbf{q}, \mathbf{q}) = 0$), this will establish our lower bound.

By Lemma 12, an ordered fingerprint from m samples from one of the k' blocks of size b gives no information in distinguishing $\mathbf{p}$ and $\mathbf{q}$. Further, it is immediate to see, from the definition of ordered fingerprints, that the following holds as well:

▷ Claim 15. Let $\mathbf{p}_1$ and $\mathbf{p}_2$ be two distributions over $[n]$, such that the distribution of fingerprints for $\mathbf{p}_b$ is given by $\{\boldsymbol{F}^b_j\}_{j\in[n]}$. Suppose there exists a partition of $[n]$ into $(I_1, I_2, \ldots, I_m)$ such that the sequences $\{\boldsymbol{F}^1_j\}_{j\in I_i}$ and $\{\boldsymbol{F}^2_j\}_{j\in I_i}$ are identically distributed for all i in $[m]$. Then $\mathbf{p}_1$ and $\mathbf{p}_2$ cannot be distinguished by ordered fingerprints.

This implies that if an algorithm sees at most m samples from every block, then it cannot distinguish between $\mathbf{p}$ and $\mathbf{q}$ with ordered fingerprints from this sample. It follows that any algorithm that distinguishes between $\mathbf{p}$ and $\mathbf{q}$ using ordered fingerprints must receive at least $m + 1$ samples from at least one of the k' many blocks.

To conclude, note that the distribution over the blocks is uniform; thus, by folklore collision estimates (see also Suzuki et al. [20]), the probability that one of the k' blocks contains $m+1$ samples is at most 1/3, given $O(k'^{1-1/(m+1)})$ samples. Since $k' = k/b = k/(5m^2 2^m)$, we get the claimed lower bound. ◀

Setting $m = \sqrt{\log k}$ in Theorem 14, we get:

▶ **Corollary 16.** *There exists an absolute constant $\varepsilon_0 > 0$ such that, for every sufficiently large integers $k \geq 1$, there exists $\mathbf{q}$ over $[k$ for which testing* **Identity-Up-To-Binning**$(k, \mathbf{q}, k, \varepsilon_0)$ *from ordered fingerprints only requires $\frac{k}{2^{O(\sqrt{\log k})}}$ samples.*

References

1 Sivaraman Balakrishnan and Larry Wasserman. Hypothesis testing for high-dimensional multinomials: a selective review. *Ann. Appl. Stat.*, 12(2):727–749, 2018. doi:10.1214/18-AOAS1155SF.

2 Tuğkan Batu, Eldar Fischer, Lance Fortnow, Ravi Kumar, Ronitt Rubinfeld, and Patrick White. Testing random variables for independence and identity. In *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 442–451. IEEE Computer Soc., Los Alamitos, CA, 2001.

3 Tuğkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D. Smith, and Patrick White. Testing that distributions are close. In *41st Annual Symposium on Foundations of Computer Science (Redondo Beach, CA, 2000)*, pages 259–269. IEEE Comput. Soc. Press, Los Alamitos, CA, 2000. doi:10.1109/SFCS.2000.892113.

4 Eric Blais, Clément L. Canonne, and Tom Gur. Distribution testing lower bounds via reductions from communication complexity. *ACM Trans. Comput. Theory*, 11(2):Art. 6, 37, 2019. doi:10.1145/3305270.

5 Clément L. Canonne. A survey on distribution testing: Your data is big. but is it blue? *Electronic Colloquium on Computational Complexity (ECCC)*, 22:63, 2015. To appear as a graduate student survey in Theory of Computing. URL: http://eccc.hpi-web.de/report/2015/063.

6 Clément L. Canonne. Problem 96: Identity testing up to coarsenings, 2019. Asked at the 3rd Workshop on Local Algorithms (WOLA 2019). URL: https://en.wikipedia.org/w/index.php?title=Plagiarism&oldid=5139350.

7 Siu-On Chan, Ilias Diakonikolas, Rocco A. Servedio, and Xiaorui Sun. Learning mixtures of structured distributions over discrete domains. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1380–1394. SIAM, Philadelphia, PA, 2012.

8 David Conlon, Jacob Fox, and Benny Sudakov. Hypergraph Ramsey numbers. *J. Amer. Math. Soc.*, 23(1):247–266, 2010. doi:10.1090/S0894-0347-09-00645-6.

9 Luc Devroye and Gábor Lugosi. *Combinatorial methods in density estimation*. Springer Series in Statistics. Springer-Verlag, New York, 2001. doi:10.1007/978-1-4613-0125-7.

10 Ilias Diakonikolas, Themis Gouleakis, John Peebles, and Eric Price. Sample-optimal identity testing with high probability. In *45th International Colloquium on Automata, Languages, and Programming*, volume 107 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages Art. No. 41, 14. Schloss Dagstuhl. Leibniz-Zent. Inform., Wadern, 2018.

11 Ilias Diakonikolas and Daniel M. Kane. A new approach for testing properties of discrete distributions. In *57th Annual IEEE Symposium on Foundations of Computer Science—FOCS 2016*, pages 685–694. IEEE Computer Soc., Los Alamitos, CA, 2016.

12 Ilias Diakonikolas, Daniel M. Kane, and Vladimir Nikishkin. Optimal algorithms and lower bounds for testing closeness of structured distributions. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science—FOCS 2015*, pages 1183–1202. IEEE Computer Soc., Los Alamitos, CA, 2015. doi:10.1109/FOCS.2015.76.

13 Ilias Diakonikolas, Daniel M. Kane, and Vladimir Nikishkin. Testing identity of structured distributions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1841–1854. SIAM, Philadelphia, PA, 2015. doi:10.1137/1.9781611973730.123.

14 Oded Goldreich. The uniform distribution is complete with respect to testing identity to a fixed distribution. In *Computational Complexity and Property Testing*, volume 12050 of *Lecture Notes in Computer Science*, pages 152–172. Springer, 2020.

15 Oded Goldreich, Shafi Goldwasser, and Dana Ron. Property testing and its connection to learning and approximation. *JACM*, 45(4):653–750, July 1998.

16 Oded Goldreich and Dana Ron. On testing expansion in bounded-degree graphs. Technical Report TR00-020, Electronic Colloquium on Computational Complexity (ECCC), 2000.

17 Andy Nguyen. Solving cyclic longest common subsequence in quadratic time. *CoRR*, abs/1208.0396, 2012. arXiv:1208.0396.

18 Liam Paninski. A coincidence-based test for uniformity given very sparsely sampled discrete data. *IEEE Trans. Inform. Theory*, 54(10):4750–4755, 2008. doi:10.1109/TIT.2008.928987.

19 Ronitt Rubinfeld and Madhu Sudan. Robust characterization of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):252–271, 1996.

20 Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In *Information security and cryptology—ICISC 2006*, volume 4296 of *Lecture Notes in Comput. Sci.*, pages 29–40. Springer, Berlin, 2006. doi:10.1007/11927587_5.

21 Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM J. Comput.*, 46(1):429–455, 2017. doi:10.1137/151002526.

Chernoff Bound for High-Dimensional Expanders

Tali Kaufman
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
kaufmant@mit.edu

Ella Sharakanski
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
Ella10004@gmail.com

Abstract

We generalize the expander Chernoff bound to high-dimensional expanders. The expander Chernoff bound is an essential property of expanders, first proved by Gillman [9]. Given a graph G and a function f on the vertices, it states that the probability of f's mean sampled via a random walk on G to deviate from its actual mean, has a bound that depends on the spectral gap of the walk and decreases exponentially as the walk's length increases.

We are interested in obtaining an analog Chernoff bound for high order walks on high-dimensional expanders. A naive generalization of the expander Chernoff bound from expander graphs to high-dimensional expanders gives a very poor bound due to obstructions that occur in high-dimensional expanders and are not present in (one-dimensional) expander graphs. Because of these obstructions, the spectral gap of high-order random walks is inherently small.

A natural question that arises is how to get a meaningful Chernoff bound for high-dimensional expanders. In this paper, we manage to get a strong Chernoff bound for high-dimensional expanders by looking beyond the spectral gap.

First, we prove an expander Chernoff bound that depends on a notion that we call the "shrinkage of a function" instead of the spectral gap. In one-dimensional expanders, the shrinkage of any function with zero-mean is bounded by $\lambda(M)$. Therefore, the spectral gap is just the one-dimensional manifestation of the shrinkage.

Next, we show that in good high-dimensional expanders, the shrinkage of functions that "do not come from below" is good. A function does not come from below if from any local point of view (called "link") its mean is zero.

Finally, we prove a high-dimensional Chernoff bound that captures the expansion of the complex. When the function on the faces has a small variance and does not "come from below", our bound is better than the naive high-dimensional expander Chernoff bound.

2012 ACM Subject Classification Theory of computation → Random walks and Markov chains; Mathematics of computing → Spectra of graphs; Mathematics of computing → Hypergraphs

Keywords and phrases High Dimensional Expanders, Random Walks, Tail Bounds

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.25

Category RANDOM

1 Introduction

Expander graphs have been studied extensively for several decades. Essentially, these are graphs that are sparse yet highly connected. They have countless applications in computer science, mathematics and physics (see e.g. the surveys of [13] and [19]).

Since expanders are such a marvelous tool, it is believed that their generalization to higher dimensions is promising. Expanders consist of edges, which are sets of two vertices. In contrast, high-dimensional (abbreviated as HD) expanders consist of faces, which are sets of any amount of vertices, like edges, triangles, pyramids, and more. Since HD expanders expand at *all dimensions*, they are much stronger objects than expander graphs, which only expand at one dimension.


© Tali Kaufman and Ella Sharakanski;
licensed under Creative Commons License CC-BY



Recently, HD expanders have attracted a lot of attention. A notable example is the paper of [1] which solves a major open problem on counting the bases of matroids. The algorithms and proofs in their paper build on recent results about *random walks on HD expanders* (e.g. [16] and [15]).

In this paper, we study further aspects of random walks on HD expanders. In particular, we generalize the expander Chernoff bound to HD expanders. Generally speaking, this bound states that a random walk on an expander acts "as expected".

Let us present the (one-dimensional) expander Chernoff bound in a more precise way. There are many forms for this bound (see e.g. [10, 14, 18, 21, 12, 20, 5]), the form we present appears in [10] and is considered standard.

Let G be an undirected connected graph on vertices $V \coloneqq \{1, \ldots, n\}$ with positive weights on its edges. The graph is equivalent to a connected random walk with transition matrix M. Let π be the stationary distribution of M and let $\{w_i\}_{i=1}^{t}$ be a random walk according to M with initial distribution π. Let $f : V \to [0, 1]$ be a function on the vertices.

The general form of the graph generalization of Chernoff bound is as follows: for all $0 \leq \epsilon \leq 1$,

$$\mathbb{P}\Big[\big|\frac{1}{t}\sum_{i=1}^{t} f(w_i) - \mathbb{E}_{\pi}[f]\big| \geq \epsilon\Big] \leq 4 \cdot \exp\big(-\Omega(\epsilon^2 st)\big). \tag{1}$$

Note that for a better bound, we want s to be bigger.

The expander Chernoff bound states that inequality (1) holds for $s = 1 - \lambda_2(M)$, where $\lambda_2(M)$ is the second largest eigenvalue of M. This bound works best for graphs with λ_2 as close to 0 as possible. By Definition 7, these are exactly (one-sided) expander graphs.

As thoroughly explained in [21], this bound is a very useful tool in theoretical computer science. It has applications in a variety of areas like construction of efficient error-correcting codes, hardness of approximation, deterministic amplification, security amplification in cryptography, extractor construction, and more.

Since high-dimensional expanders are very promising, and the Chernoff bound for expander graphs is fundamental and extremely useful, we see great potential in further generalizing it to higher dimensions.

Such a generalization should provide a Chernoff bound for random walks on high-dimensional faces, that is similar to the bound we have for walks on vertices. Since high-dimensional expanders are better than one-dimensional, one could expect to obtain a generalization simply by applying the existing expander Chernoff bound to high-order walks.

However, this method results in a surprisingly poor bound, due to natural obstructions that occur even in the best high-dimensional expanders, and do not happen for one-dimensional expanders. Because of these obstructions, the spectral gap $1 - \lambda(M)$ of high-order random walks is inherently small.

In this paper, we manage to overcome this problem by looking beyond the spectral gap. First, we prove an expander Chernoff bound that depends on the "shrinkage of the function" f (see Definition 3) instead of the spectral gap. In one-dimensional expanders, the shrinkage of any function with zero-mean is bounded by $\lambda(M)$. Therefore, the spectral gap is just the one-dimensional manifestation of the shrinkage.

Next, we show that in good high-dimensional expanders, the shrinkage of functions that "do not come from below" (see Definition 1) is good. A function does not come from below if its mean is zero from any local point of view (called "link", see Definition 11). This criterion naturally generalizes zero-mean functions since there is only one local point of view on vertices and it includes all of them.

Thus, we obtain a high-dimensional expander Chernoff bound that works well for functions that have a small variance and do not come from below.

1.1 Background

Let us provide a simplified version of the definitions we need in order to present our results. The detailed definitions are deferred to Section 2.

Given a weighted undirected connected graph G, its normalized second largest eigenvalue in absolute value is denoted $\lambda(G)$. We say that G is a (two-sided) λ-*spectral expander* if $\lambda(G) < \lambda$.

We study a generalization of the above definition to higher dimensions, based on *simplicial complexes*. A pure d-dimensional simplicial complex X consists of sets of size $d+1$ and all of their subsets. These sets are called faces, and $X(i)$ denotes the set of faces of size $i+1$. Our complexes are equipped with a probability distribution π_d over the d-faces, which induces probability distributions π_k over k-faces. In addition, each π_k induces an inner-product on the subspace of k-cochains (functions from $X(k)$ to $\mathbb{R}$) denoted $C^k(X)$.

There are several different ways to define high-dimensional expanders. We use a notion called (two-sided) local-spectral expander, defined in [7], since it is the generalization of spectral-expanders to higher dimensions. Given a face $\tau \in X(i)$, the *link* of τ, denoted X_τ, is its local point of view. Formally, it is a simplicial complex obtained by taking the faces of X that contain τ and removing τ from them. A pure complex is a γ-*local-spectral expander* if for any link, the underlying graph is a γ-spectral expander (see Definition 13).

The high-order random walk we study is called the lazy upper random walk on k-faces, defined in [15] and denoted M_k^+. This walk naturally generalizes the standard lazy walk on graphs. It can move between two k-faces if they are both contained in the same $(k+1)$-face.

Given a function $f : X(k) \to \mathbb{R}$ and a face $\tau \in X(j)$, the localization of f on X_τ is the function $f_\tau : X_\tau(k-j-1) \to \mathbb{R}$ defined by $f_\tau(\sigma) = f(\tau \cup \sigma)$.

▶ **Definition 1.** *We say that a function $f : X(k) \to \mathbb{R}$ is* from level $j+1$ *if its expectation on the links of j-faces is zero. Formally, f is from level $j+1$ if $\mathbb{E}_{\pi_\tau}[f_\tau] = 0$ for all $\tau \in X(j)$.*

Informally, we say that f *does not come from below* if it is from a high level (e.g. from level k). Our high-dimensional Chernoff bound works best for functions f that do not come from below and have a small variance.

1.2 Our Contribution: Approach and Considerations

A bound for high-order random walks. Observe that the graph Chernoff bound is actually a bound for *random walks* on graphs. Hence, what we are looking for is a bound for *random walks* on high-dimensional expanders. Specifically, we study the lazy upper random walk, since it is highly related to the structure of the HD expander (see Definition 17).

Capturing the expansion. Given a HD γ-local-spectral expander X (see Definition 13), we want a bound that captures the expansion of X. We can apply the graph Chernoff bound (1) to the random walk M_k^+ and get a bound where $s = 1 - \lambda(M_k^+)$ (see Definitions 7). Unfortunately, this value of s is bounded away from 1 (in fact, it behaves like $1/k$, so it approaches 0 as k increases) and hence does not capture the expansion of X well.

According to [17], $\lambda(M_k^+)$ is large due to natural obstructions that occur in HD expanders and not in one-dimensional expander graphs. To bypass the obstructions, our Chernoff bound takes into account the entire spectrum of M_k^+.

Generalizing a one-dimensional Chernoff bound. We prove our high-dimensional Chernoff bound by generalizing a one-dimensional Chernoff bound. Given an expander graph G and a function f on the vertices (as in (1)), our key idea is to only look at eigenvalues λ_i that correspond to eigenvectors that are part of the orthogonal decomposition of f into eigenspaces. We prove a Chernoff bound that depends on the shrinkage of f, denoted $\lambda(f) = \max_i|\lambda_i|$ (see Definition 3), instead of $\lambda(G)$. When f is "good" (not an obstruction), $\lambda(f)$ is a small eigenvalue of G. By applying our one-dimensional Chernoff bound to M_k^+, we obtain a high-dimensional Chernoff bound that can see beyond $\lambda(M_k^+)$ and therefore capture the expansion of X well.

A bound for functions that do not come from below. The standard one-dimensional expander Chernoff bound depends on $\lambda(G)$, which is the largest eigenvalue in absolute value except for 1. Namely, the bound only considers eigenspaces orthogonal to $\text{span}\{\mathbf{1}\}$. Equivalently, it considers eigenspaces of zero-mean functions. These are exactly functions that their expectation on the links of (-1)-faces is zero (since the only (-1)-face is the empty face and its link is the entire set of vertices), which we call functions from level 0. Therefore, the existing bound is in fact an expander Chernoff bound for functions from level 0. Similarly, our HD expander Chernoff bound applies to functions from level j.

Moreover, as we will prove in Corollary 37, given a good enough γ-local-spectral expander, a function is from level $j+1$ if and only if its orthogonal decomposition into eigenspaces of M_k^+ only includes eigenvectors matching eigenvalues smaller than $\frac{k-j}{k+2} + o_\gamma(1)$. Namely, a function does not come from below if and only if its shrinkage is small. Hence, considering functions that do not come from below is highly natural.

Irregularity matters. Some expander Chernoff bounds apply only to regular graphs (see e.g. [12, 8]). In the case of HD expanders, even when the weights of the largest faces are uniform, the weights of other k-dimensional faces are usually not uniform, and hence M_k^+ is usually irregular. Therefore, it is important that our bound applies to irregular graphs too.

1.3 Results and Technique

Our main result is a Chernoff bound for HD expanders. When the function on the k-faces has a small variance and does not "come from below", our bound captures the expansion of the complex well. Namely, we present a Chernoff bound for M_k^+ in the form of (1), where s approaches 1 as k increases, as opposed to the naive bound where s approaches 0.

▶ **Theorem 2** (Main theorem, HD expander Chernoff bound, informal, for formal see Theorem 48)**.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \le k < d$, and assume that γ is small enough. Let $\pi = \pi_k$ be the probability distribution over $X(k)$ and denote $\pi_* := \min \pi_i$. Let $w_1, \ldots, w_t$ be a random walk according to M_k^+ with initial distribution π. Let $f : X(k) \to \mathbb{R}$ such that f is from level $j+1$ for some $-1 \le j < k$, and assume $\|f\|_\infty = 1$. Then the following hold:*

1. *If $\text{Var}_\pi(f) \le \pi_* \epsilon^2 c$ for a constant $c \in \mathbb{R}$ then inequality (1) holds for $s = \left(1 - \frac{k-j}{k+2} + o_\gamma(1)\right)^2$. In particular, for $j = k-1$, inequality (1) holds for $s = \left(1 - \frac{1}{k+2} + o_\gamma(1)\right)^2$.*
2. *If $\text{Var}_\pi(f) \le \pi_* \epsilon^2 c$ such that $4c \le \left(\frac{k+2}{k-j+o_\gamma(1)}\right)^2$, then inequality (1) holds for $s = 1$.*

   In particular, for $j = k-1$, if $c \le \frac{(k+2)^2}{4+o_\gamma(1)}$ then inequality (1) holds for $s = 1$.

Note that $o_\gamma(1)$ refers to a value that approaches 0 as γ goes to 0.

Extensions of the theorem. In the detailed version of this theorem, stated in Theorem 48, we also provide a bound for larger $\mathrm{Var}_\pi(f)$. Additionally, in Remark 49 we provide a bound for cases where f does come from below, but its mass on the lower eigenspaces is small.[1]

Comparison with the naive bound. One could try to achieve a similar bound by applying to M_k^+ a standard expander Chernoff bound like [10]. However, this standard bound depends on the spectral gap, which is low even in the best HD expanders. This method would give $s \approx 1/(k+2)$, which approaches 0 as k increases. We successfully overcome this problem by considering the shrinkage of f in functions that do not come from below, instead of the spectral gap. Indeed, we achieve a value of s that is either 1 or it approaches 1 as k grows.

Proof overview. To prove this theorem, we first define the *shrinkage* of a vector by an operator (Definition 3). The shrinkage is key to overcoming the obstructions which make $\lambda(M_k^+)$ large. We then prove a (one-dimensional) expander Chernoff bound that depends on the shrinkage of the function on the vertices, instead of on the graph's expansion (Theorem 4). Next, we apply to M_k^+ our expander Chernoff bound for shrinking functions. Finally, we bound the shrinkage of k-cochains (Proposition 5) to prove the HD expander Chernoff bound.

1.3.1 Defining the Shrinkage of a Vector

Let W be a finite-dimensional vector-space with inner-product $\langle \cdot, \cdot \rangle$. Let $M : W \to W$ be a self-adjoint operator. By the spectral theorem, $W = \bigoplus_i E_{\lambda_i}$ where E_{λ_i} is the eigenspace of M associated with the eigenvalue $\lambda_i \in \mathbb{R}$. Moreover, there exists an orthonormal basis of W that consists of eigenvectors of M.

▶ **Definition 3.** *Let $w \in W$. The* shrinkage *of w is $\lambda_M(w) \coloneqq \max_i\{|\lambda_i| \mid w \not\perp E_{\lambda_i}\}$. Equivalently, $\lambda_M(w)$ is the largest eigenvalue of M (in absolute value) that matches an eigenvector that spans w. Formally, let B be any orthonormal basis of W consisting of eigenvectors ϕ_i corresponding to eigenvalues λ_i respectively. Then $\lambda_M(w) \coloneqq \max_i\{|\lambda_i| \mid w \not\perp \phi_i\}$. For ease of notation, when M is clear from context, we simply write $\lambda(w)$.*

1.3.2 Expander Chernoff Bound for Shrinking Functions

We prove an expander Chernoff bound that depends on the shrinkage of the function on the vertices, $\lambda(f)$, instead of on the graph's expansion, $\lambda(G)$. As far as we are aware, this is the first bound of this kind. The bound is a generalization of a similar bound from [8].

▶ **Theorem 4** (Expander Chernoff bound that depends on $\lambda(f)$ instead of $\lambda(G)$, informal, for formal see Theorem 29)**.** *In the setting of inequality (1), assume the function $f : V \to \mathbb{R}$ satisfies $\mathbb{E}_\pi[f] = 0$ and $\|f\|_\infty = 1$, and denote $\pi_* \coloneqq \min \pi_i$. Then the following hold:*

1. *If $\mathrm{Var}_\pi(f) \le \pi_* \epsilon^2 c$ for some constant $c \in \mathbb{R}$ then inequality (1) holds for $s = (1 - \lambda(f))^2$.*
2. *Moreover, if $\mathrm{Var}_\pi(f) \le \pi_* \epsilon^2 c$ such that $c \le \dfrac{1}{4\lambda(f)^2}$ then inequality (1) holds for $s = 1$.*

In the detailed version of this theorem, stated in Theorem 29, we also provide a bound for when $\mathrm{Var}_\pi(f)$ is larger.

[1] Namely, denote $P_{k,j} \coloneqq P_{(\mathrm{Im}\, d_{j \nearrow k})^\perp}$ the orthogonal projection on the upper subspaces and $\ell \coloneqq \|f - P_{k,j} f\|_\pi$. Then the bound above holds if $\ell < \epsilon\sqrt{\pi_*}/2$ and $\mathrm{Var}_\pi(f) \le (\epsilon\sqrt{\pi_*} - 2\ell)^2 c$.

The proof is by showing a generic reduction from the problem of concentration for random walks to the well-studied problem of concentration for martingales (see e.g. [4]). This reduction is a generalization of [8, Theorem 1.6] and it may be of independent interest. As in [8], we then use Azuma's inequality to finish the proof.

1.3.3 Bounding the Shrinkage of k-cochains

The following proposition shows that functions that do not "come from below" shrink well, thereby relating the structure of a complex and its spectral properties.

▶ **Proposition 5** (Shrinkage of k-cochains, informal, for formal see Proposition 41)**.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \leq k < d$, and assume γ is small enough. Let $f : X(k) \to \mathbb{R}$ such that f is from level $j+1$ for some $-1 \leq j < k$. Then*

$$\lambda_{M_k^+}(f) \leq \frac{k-j}{k+2} + o_\gamma(1).$$

We prove the proposition by showing that the spectral decomposition (see Definition 24) and the combinatorial decomposition (see Definition 23) of $C^k(X)$, both originally defined in [17], are identical (see Theorem 36).

Furthermore, we calculate the number of functions that do not come from below and hence shrink well (see Corollary 45), to show that they are very common.

1.4 Related Work and Comparison

First of all, let us compare our expander Chernoff bound, stated in Theorem 29, to other related results. Compared to the bound from [8, Theorem 1.6], our bound improves and generalizes their proof in a few ways. First, our proof applies to any stationary distribution, not necessarily a uniform one. Secondly, our bound depends on $\lambda(f)$ (see Definition 3) instead of $\lambda(G)$. Thirdly, we provide a better bound for the case where $\mathrm{Var}_\pi(f)$ is small enough. Lastly, we maintain the dependence of the bound on $\mathrm{Var}_\pi(f)/\pi_*$ instead of bounding it by n. Note that the proof of [8] applies to vector-valued functions $f : V \to \mathbb{R}^N$, but we only prove the case of $N = 1$ for simplicity.

Compared to other similar bounds, the first advantage of our bound is that it depends on $\lambda(f)$ instead of $\lambda(G)$. Secondly, our proof has the advantage of being elementary and not use perturbation theory. Lastly, our bound includes a special case concerning $\mathrm{Var}_\pi(f) \leq \pi_* \epsilon^2 / 4\lambda(f)^2$, in which the bound does not depend on the spectral gap at all and therefore it is better than any other bound that we know of.

Since our bound depends on $\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*)$, it is a better fit for cases where $\mathrm{Var}_\pi(f)$ is small enough compared to ϵ (or, equivalently, ϵ is large enough), and where the stationary distribution is close to being uniform.

For example, compared to [10] (see (1)), the main advantage of our bound is that it depends on $(1-\lambda(f))^2$, which can sometimes be much larger than $1-\lambda_2(G)$. Moreover, our proof is elementary whereas their proof is based on perturbation theory. However, their bound does not depend on $\mathrm{Var}_\pi(f)$ at all, so it is better when $\mathrm{Var}_\pi(f)/\pi_*$ is large compared to ϵ.

Compared to [18, Theorem 1.1], both bounds are influenced by the variance of f, but our bound depends on $\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*)$ whereas their bound depends on $\|f\|_2^2$. So for most values of $\|f\|_2$, π_* and ϵ, e.g. when $\|f\|_2 > 8$, $\pi_* = 1/|V|$ and $\epsilon > 0.01$, our bound is better. Other than that, as with [10], their bound has $1-\lambda_2(G)$ while our bound has $(1-\lambda(f))^2$, and their proof is based on the non-elementary perturbation theory while our method has the advantage of being elementary.

Compared to [18, Section 4], their bound depends on $\|f\|_2^2$, whereas our bound depends on $\ln(4\operatorname{Var}_\pi(f)/\epsilon^2\pi_*)$, which is mostly better when ϵ is not too small. Moreover, their bound is only applicable for $\epsilon \le \|f\|_2^2$. Lastly, as before, their bound depends on $1-\lambda(G)$, and our bound depends on $(1-\lambda(f))^2$ which can be significantly smaller.

Secondly, let us compare our Chernoff bound for high-dimensional expanders, stated in Theorem 48, to related results. As far as we know, there are no existing high-dimensional Chernoff bounds. However, we can still compare our bound to the bounds we get by applying the existing (one-dimensional) graph Chernoff bounds to M_k^+. This comparison is covered in depth by the above paragraphs. In short, if f has a small enough variance and does not "come from below", our bound is better since it depends on $\lambda(f)$ instead of $\lambda(M_k^+)$ or $\lambda_2(M_k^+)$. Note that since all of the eigenvalues of M_k^+ are non-negative, $\lambda(f) \le \lambda(M_k^+) = \lambda_2(M_k^+)$.

Finally, our work in Section 5 improves some results from [17]. Corollary 39 improves [17, Theorem 5.9] (stated here as Proposition 19). It shows that the set containing the spectrum of M_k^+ is even smaller than what was known before. As a consequence, corollary 40, which gives an upper bound on $\|(M_k^+)^i\phi\|$, improves [17, Corollary 5.11].

1.5 Organization

This paper is organized as follows. Section 2 provides the relevant background in detail. In Section 3 we study the shrinkage of a vector denoted $\lambda(w)$. In Section 4 we provide a Chernoff bound for expander graphs which depends on the shrinkage of the function on the vertices. In Section 5 we study the shrinkage of vectors by high-order random walks and also improve some existing results. In Section 6 we prove our main result regarding Chernoff bound for high-dimensional expanders. In Appendix A we prove the the bound from Section 4. Due to space restrictions, some of the proofs are omitted and can be found in the full version of the paper.

2 Preliminaries

In this section, we present the relevant background about expander graphs, high-dimensional expanders, random walks, and decompositions of the subspace of k-cochains. We provide more details in the full version of this paper.

2.1 Expander Graphs and Random Walks

In this subsection, we define walk operators and the inner product-space they act on. Then, we define λ-spectral expander graphs.

In this paper, we will consider only finite state-spaces and only *time-homogeneous* Markov chains. A *random walk* M is a finite reversible Markov chain. A sequence of random variables according to M is also called random walk. Throughout this paper, for ease of presentation and connection with graphs, we say that a random walk is *connected* instead of irreducible. By the book of [11], a connected random walk has a positive and unique stationary distribution.

▶ **Definition 6** (Walk operator, $l_2(\pi)$)**.** *Let M be a connected random walk over state-space V and let π be its stationary distribution. We denote by $l_2(\pi)$ the vector-space of functions $f : V \to \mathbb{R}$ with inner product $\langle f,g\rangle_\pi = \sum_{i\in V}\pi(i)f(i)g(i)$. It is indeed an inner product since π is positive. The* walk operator *$M : l_2(\pi) \to l_2(\pi)$ is defined by $(Mf)(i) \coloneqq \sum_{j\in V} M(i,j)f(j)$.*

It is easy to see that a connected random walk operator M is self-adjoint. Therefore, by the spectral theorem, there is an orthonormal basis of $l_2(\pi)$ consisting of eigenvectors of M, and each eigenvalue of M is real. We are now ready to define expanders:

▶ **Definition 7** (λ-spectral expander graph). *Let G be an undirected connected graph with n vertices and let $\lambda_1 \geq \cdots \geq \lambda_n$ be its (real) eigenvalues. G is a* λ-spectral expander *if $\lambda(G) \leq \lambda$ where $\lambda(G) \coloneqq \max\{|\lambda_2|, |\lambda_n|\}$. Alternatively, G is a* one-sided λ-spectral expander *if $\lambda_2 \leq \lambda$.*

A small value of λ means that the graph is a better expander. λ also controls how well M shrinks functions, since $\|Mf\|_\pi \leq \lambda\|f\|_\pi$ for any f such that $\mathbb{E}_\pi[f] = 0$ (again by the spectral theorem). Therefore good expanders *shrink functions* well. The following proposition is well-known:

▶ **Proposition 8.** *Let G be an undirected connected graph. G is non-bipartite if and only if $\lambda(G) < 1$.*

2.2 High-Dimensional Expanders

High-dimensional expanders are a special type of abstract simplicial complexes. They are similar to graphs, but they consist of faces instead of just edges. Faces can be edges, but they can also be vertices, triangles, pyramids, etc. Let us provide the formal definitions.

A *face* τ is a set of vertices. The *dimension* of τ is defined by $\dim(\tau) \coloneqq |\tau| - 1$. Let X be a set of faces. X is an *abstract simplicial complex* if for every face $\tau \in X$, if $\sigma \subset \tau$ then $\sigma \in X$. We denote by $X(k)$ the set of faces of dimension $k \geq -1$. The *dimension* of X is $\max\{\dim(\tau) \mid \tau \in X\}$.

▶ **Definition 9** (Pure simplicial complex). *Let X be a d-dimensional abstract simplicial complex. X is called* pure *if for every $\sigma \in X$, there is a d-dimensional face τ such that $\sigma \subset \tau$.*

As a generalization of weighted graphs, we have weighted simplicial complexes. The weight of smaller faces is induced by the weight of larger faces to comply with the high-dimensional structure.

▶ **Definition 10** (Weighted simplicial complex). *Let X be a pure d-dimensional simplicial complex. Let π_d be a positive probability distribution over its d-dimensional faces. For each $-1 \leq k \leq d$, the induced probability distribution over the k-faces is denoted π_k, where $\pi_k(\sigma)$ is the probability to choose a d-face τ according to π_d and then get σ when uniformly choosing a k-face contained by τ. A pure simplicial complex equipped with a distribution is called a* weighted simplicial complex.

There are a few related but not equivalent definitions of HD expanders. Each one of them generalizes a different aspect of expander graphs. We stick to the definition that is related to random walks, called *local-spectral expander*. Basically, it says that from any local point of view (or *link*), the complex should be expanding.

▶ **Definition 11** (Link). *Let X be a pure d-dimensional simplicial complex. Let $\tau \in X(k)$ be a k-dimensional face. The* link *of τ, denoted X_τ, is a pure $(d-k-1)$-dimensional simplicial complex obtained by taking every face that contains τ and removing τ's vertices from it. Formally speaking, $X_\tau \coloneqq \{\sigma \setminus \tau \mid \tau \subseteq \sigma \in X\}$.*

If X is weighted, we give each top face in X_τ a probability proportional to its probability in X, normalized so that the sum of the probabilities is 1. The probability is well-defined since every face in X_τ is also a face in X.

▶ **Definition 12** (Skeleton). *Let X be an abstract simplicial complex. The* skeleton *of X, also called the* underlying graph*, is an undirected graph with $X(0)$ as its vertices and $X(1)$ as its edges. If X is weighted, the skeleton inherits its weights.*

▶ **Definition 13** (High-dimensional expander). *Let X be a pure complex. If the skeleton of every one of its links is a γ-spectral expander then X is a* γ-local-spectral expander*.*

▶ **Definition 14** (Degree). *Let X be an abstract simplicial complex and let $v \in X(0)$ be a vertex in X. The* degree *of v is the number of faces that contain it, namely, $|X_v|$. The* degree *of X is* $\max_{v \in X(0)} |X_v|$.

As with expander graphs, we are interested in infinite families of d-dimensional bounded degree γ-local-spectral expanders. Explicit constructions of such families exist, see [7, Lemma 1.5] for example. Moreover, we know that explicit constructions with a D-flat top distribution exist:

▶ **Definition 15** (D-flat). *Let π be a probability distribution over V and let $D \in \mathbb{N}$. We say that π is D-flat if there exists some $R \in \mathbb{R}$ such that $\pi_v \in \{\frac{1}{R}, \frac{2}{R}, \ldots, \frac{D}{R}\}$ for all $v \in V$.*

▶ **Proposition 16** ([6, Claim 6.2]). *For every $\gamma > 0$ and $d \in \mathbb{N}$ there exists an explicit infinite family of bounded degree d-dimensional complexes which are γ-local-spectral expanders. They have top distribution π_d that is D-flat, for $D \leq (1/\gamma)^{O(d^2/\gamma^2)}$.*

2.2.1 Random Walks on High-Dimensional Expanders

One can define many different random walks on a given HD expander. We are specifically interested in a walk that captures the expanding structure of the complex. Such walks were defined for the first time by [15].

▶ **Definition 17** (Lazy upper random walk). *Let X be a weighted pure d-dimensional simplicial complex. Let $0 \leq k < d$. Define M_k^+ to be the* lazy upper random walk *on the k-dimensional faces: For each $\sigma, \tau \in X(k)$, $M_k^+(\sigma, \tau)$ is the probability to choose a $(k+1)$-face $\eta \supset \sigma$ according to π_{k+1} and then get τ when uniformly choosing a k-face contained by η. Equivalently, $M_k^+ : X(k) \times X(k) \to \mathbb{R}$ is:*

$$M_k^+(\sigma,\tau) := \begin{cases} \frac{1}{k+2} & \sigma = \tau \\ \frac{\pi_{k+1}(\sigma \cup \tau)}{(k+2)\pi_k(\sigma)} & \sigma \cup \tau \in X(k+1) \\ 0 & \text{otherwise} \end{cases}$$

This walk is called lazy *since $M(\sigma, \sigma) > 0$ so it considers standing-still a possible move.*

Let us provide the following definition from [17, Theorem 5.6]:

▶ **Definition 18.** *Let X be a weighted pure d-dimensional γ-local-spectral expander. Define $\epsilon_0 = \gamma$ and $\epsilon_k = 2k(1 + 2k\sqrt{k})\epsilon_{k-1} + (k+1)\gamma$ for all $0 < k < d$. We say that γ is* small enough with respect to d *if $\epsilon_{k-1} \leq \frac{1}{2(1+2k\sqrt{k})}$ for all $0 < k < d$, and $\epsilon_{d-1} < \frac{1}{2\sqrt{d}}$.*

Note that $\lim_{\gamma \to 0} \epsilon_k = 0$. Therefore, we can construct a γ-local-spectral expander with a small enough γ by using the explicit construction from Proposition 16 with a sufficiently small γ.

The following proposition shows that even though the second largest eigenvalue of M_k^+ goes to 1 as k increases, we can go beyond that eigenvalue to achieve much better results. For example, when γ is small enough, the smallest eigenvalue of M_k^+ goes to 0 as k increases.

▶ **Proposition 19** ([17, Theorem 5.9]). *Let X be a weighted d-dimensional γ-local-spectral expander. If γ is small enough with respect to d then for every $0 \le k < d$:*

$$\mathrm{Spec}(M_k^+) \subseteq \{1\} \cup \bigcup_{j=0}^{k} \left[\frac{k+1-j}{k+2} - \frac{\sqrt{k+1}}{k+2}\epsilon_k, \frac{k+1-j}{k+2} + \frac{\sqrt{k+1}}{k+2}\epsilon_k \right]$$

where $\mathrm{Spec}(M_k^+)$ *is the set of eigenvalues of* M_k^+.

2.2.2 Decompositions of the Subspace of k-Cochains

Let X be an d-dimensional γ-local-spectral expander. For all $-1 \le k \le d$, define $C^k(X)$ to be the subspace of k-cochains, i.e., functions $\phi : X(k) \to \mathbb{R}$. Let $\pi = \pi_k$ be the probability distribution over $X(k)$. Let the inner-product of $C^k(X)$ be $\langle \phi, \psi \rangle \coloneqq \sum_{i \in X(k)} \pi(i)\phi(i)\psi(i)$ and let $\|\cdot\|$ be the induced norm. Note that as a walk operator, $M_k^+ : C^k(X) \to C^k(X)$.

For all $-1 \le k \le d$, define $C_0^k(X) \coloneqq \{\phi \in C^k(X) \mid \phi \perp \mathbf{1}\}$ where $\mathbf{1}$ is the constant 1 function. For all $-1 \le j \le k \le d$, the operator $d_{j \nearrow k} : C^j(X) \to C^k(X)$ is defined by:

$$d_{j \nearrow k}\phi(\sigma) \coloneqq \sum_{\substack{\tau \in X(j) \\ \tau \subseteq \sigma}} \phi(\tau)$$

for all $\phi \in C^j(X)$ and $\sigma \in C^k(X)$. Note that $d_{k \nearrow k}$ is the identity function. In addition, for $k < d$, we denote $d_k \coloneqq d_{k \nearrow k+1}$.

▶ **Lemma 20** ([17, Section 5.3]). *Let $0 \le j \le k$. Then $d_{j-1 \nearrow k}(C_0^{j-1}(X)) \subseteq d_{j \nearrow k}(C_0^j(X))$.*

This operator is highly related to the lazy upper random walk:

▶ **Lemma 21** ([17, Corollary 3.7]). *$d_k^* d_k = (k+2)M_k^+$ for all $-1 \le k < d$.*

In addition, $d_{j \nearrow k}$ and d_j are related in the following way:

▶ **Lemma 22** ([17, Proposition 5.5]). *$d_{j \nearrow k} = \frac{1}{(k-j)!} d_{k-1} \dots d_j$ for all $-1 \le j < k \le d$.*

The following definition implies a combinatorial orthogonal decomposition of $C^k(X)$.

▶ **Definition 23.** *For all $0 \le j \le k \le d$, let U_k^j be a subspaces of C_0^k that contains k-cochains that come from j-cochains but not from $(j-1)$-cochains. Formally,*

$$U_k^j \coloneqq d_{j \nearrow k}(C_0^j(X)) \cap (d_{j-1 \nearrow k}(C_0^{j-1}(X)))^\perp$$

By [17, Section 5.3], the following is an orthogonal decomposition:

$$C^k(X) = \mathrm{span}\{\mathbf{1}\} \oplus U_k^k \oplus U_k^{k-1} \oplus \cdots \oplus U_k^0.$$

This decomposition is *combinatorial* in the sense that each subspace U_k^j is related to the structure of X. Let us provide the definition of another orthogonal decomposition of $C^k(X)$, which is a *spectral* decomposition, relying on Proposition 19.

▶ **Definition 24.** *For all $0 \le j \le k < d$, we define the* neighborhoods *of M_k^+ to be:*

$$W_k^j \coloneqq \mathrm{span}\left\{ \phi \;\middle|\; M_k^+\phi = \mu\phi, \mu \in \left[\frac{k+1-j}{k+2} - \frac{\sqrt{k+1}}{k+2}\epsilon_k, \frac{k+1-j}{k+2} + \frac{\sqrt{k+1}}{k+2}\epsilon_k \right] \right\}$$

If γ is small enough then these subspaces intersect trivially and hence:

$$C^k(X) = \mathrm{span}\{\mathbf{1}\} \oplus W_k^k \oplus W_k^{k-1} \oplus \cdots \oplus W_k^0.$$

3 The Shrinkage of a Vector

Let W be a finite-dimensional vector-space with inner-product $\langle \cdot, \cdot \rangle$ and induced norm $\|\cdot\|$.

Given $u, w \in W$, we say that u is perpendicular to w and denote by $u \perp w$, if $\langle u, w \rangle = 0$. Similarly, $u \not\perp w$ if $\langle u, w \rangle \neq 0$. Moreover, given a subspace U, we denote $w \perp U$ if $w \perp u$ for all $u \in U$. We denote $w \not\perp U$ if there exists a vector $u \in U$ such that $w \not\perp u$.

Let $M : W \to W$ be a self-adjoint operator. By the spectral theorem, there exists an orthonormal basis of W that consists of eigenvectors of M.

▶ **Lemma 25.** *Let $w \in W$ and $k \geq 0$. Then $\lambda(M^k w) = \lambda(w)$.*

Proof. Let $k \geq 0$. By the definition of $\lambda(w)$ it is enough to show that $w \perp \phi \iff (M^k w) \perp \phi$ where ϕ is an eigenvector of M with eigenvalue $\lambda \neq 0$. This is true because $\langle M^k w, \phi \rangle = \langle w, M^k \phi \rangle = \langle w, \lambda^k \phi \rangle = \lambda^k \langle w, \phi \rangle$ since M^k is self-adjoint. ◀

The following lemmas explains why the shrinkage of a vector is called this way. They show that $\lambda(w)$ is an upper bound for how much M shrinks w.

▶ **Lemma 26.** *Let $w \in W$. Then $\|Mw\| \leq \lambda(w)\|w\|$.*

▶ **Lemma 27.** *Let $w \in W$ and $k \geq 0$. Then $\|M^k w\| \leq \lambda(w)^k \|w\|$.*

Proof. The proof is by induction on k together with Lemma 25 and Lemma 26. ◀

▶ **Lemma 28.** *Let $w \in W$ and let $B = \{\phi_i\}$ be an orthogonal basis of eigenvectors of M matching eigenvalues λ_i respectively. Let $C \subseteq B$ such that $w \in \text{span}(C)$. Then $\lambda(w) \leq \max_{\phi_i \in C} |\lambda_i|$.*

Proof. By definition $\lambda(w) = |\lambda_j|$ for some index j such that $w \not\perp \phi_j$. This implies that $\phi_j \in C$ since $w \in \text{span}(C)$. Therefore $\lambda(w) = |\lambda_j| \leq \max_{\phi_i \in C} |\lambda_i|$. ◀

4 Expander Chernoff Bound for Shrinking Functions

In the following section, we show an expander Chernoff bound that depends on the shrinkage of the function on the vertices $\lambda(f)$, instead of the graph's expansion. It allows us to see beyond the spectral gap and instead consider the entire spectrum of the expander. As far as we are concerned, it is the only bound of this kind. The proof is deferred to the Appendix.

▶ **Theorem 29** (Expander Chernoff bound that depends on $\lambda(f)$ instead of $\lambda(M)$)**.** *Let $M : l_2(\pi) \to l_2(\pi)$ be a connected non-bipartite random walk operator over state-space $V = \{1, \ldots, n\}$ with a stationary distribution π. Let $w_1, \ldots, w_t$ be a random walk according to M with initial distribution π and denote $\pi_* := \min_i \pi_i$. Let $f \in l_2(\pi)$ be a function satisfying $\mathbb{E}_\pi[f] = 0$. Then for every $\epsilon > 0$:*

$$\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \leq \begin{cases} 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2}\right) & \text{Var}_\pi(f) \leq \frac{\epsilon^2 \pi_*}{4\lambda(f)^2} \\ 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2 \mathcal{L}^2}\right) & \textit{otherwise} \end{cases}$$

where $\mathcal{L} := \left\lceil \frac{\ln(4\,\text{Var}_\pi(f)/\epsilon^2 \pi_*)}{1-\lambda(f)} \right\rceil$.

▶ Remark 30. We can prove a more general version of Theorem 29, appropriate for functions that have a small mass on eigenspaces associated with a large eigenvalue. The proof is the same, except we use the generalized version of Proposition 52 which appears in Remark 62. Define P as the orthogonal projection on eigenspaces associated with small eigenvalues, and denote $\ell := \|f - Pf\|_\pi$. If $\ell < \epsilon\sqrt{\pi_*}/2$, then we get the following expander Chernoff bound. For all $\epsilon > 0$:

$$\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \leq \begin{cases} 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2}\right) & \mathrm{Var}_\pi(f) \leq \left(\frac{\epsilon\sqrt{\pi_*} - 2\ell}{2\lambda(Pf)}\right)^2 \\ 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2\mathcal{L}^2}\right) & \text{otherwise} \end{cases}$$

where $\mathcal{L} := \left\lceil \frac{\ln(4\,\mathrm{Var}_\pi(f)/(\epsilon\sqrt{\pi_*} - 2\ell)^2)}{1 - \lambda(Pf)} \right\rceil$.

5 Shrinking Functions Do Not Come From Below

In this section, we discuss which functions M_k^+ shrinks well. We prove that these are exactly functions that do not come from below. In addition, we find what it means to "come from below" and how rare it is, by studying the two decompositions of $C_0^k(X)$ and the spectrum of M_k^+.

First, we provide an equivalent definition for functions from level $j+1$ (and for functions that do not come from below). The following lemmas relate functions that do not come from below to the subspace $(\mathrm{Im}\, d_{j\nearrow k})^\perp$.

▶ **Lemma 31.** *Let X be a d-dimensional γ-local-spectral expander. Let $-1 \leq j \leq k \leq d$ and $f \in C^k(X)$. Then $d^*_{j\nearrow k} f(\tau) = \mathbb{E}_{\pi_\tau}[f_\tau]$ for all $\tau \in X(j)$.*

Proof. We prove by induction on j. For $j = k$, we have that $d^*_{k\nearrow k} f(\tau) = f(\tau)$. On the other hand, $X_\tau = \varnothing$ and $\pi_\tau(\varnothing) = 1$. Therefore, $\mathbb{E}_{\pi_\tau}[f_\tau] = f_\tau(\varnothing) = f(\tau \cup \varnothing) = f(\tau) = d^*_{k\nearrow k} f(\tau)$ as needed. Next, we assume that the lemma holds for all $j < m \leq k$ and prove for j. Note that

$$\mathbb{E}_{\pi_\tau}[f_\tau] = \sum_{\sigma \in X_\tau(k-j-1)} \pi_\tau(\sigma) f_\tau(\sigma) = \sum_{\nu \in X(k), \tau \subseteq \nu} \frac{\pi(\nu)}{\pi(\tau)} f(\nu)$$

And by the induction assumption,

$$\begin{aligned} d^*_{j\nearrow k} f(\tau) &= \frac{1}{k-j} \cdot d^*_j d^*_{j+1\nearrow k} f(\tau) = \frac{1}{k-j} \cdot \sum_{\eta \in X(j+1), \tau \subseteq \eta} \frac{\pi(\eta)}{\pi(\tau)} d^*_{j+1\nearrow k} f(\eta) \\ &= \frac{1}{k-j} \cdot \sum_{\eta \in X(j+1), \tau \subseteq \eta} \frac{\pi(\eta)}{\pi(\tau)} \mathbb{E}_{\pi_\eta}[f_\eta] \\ &= \frac{1}{k-j} \cdot \sum_{\eta \in X(j+1), \tau \subseteq \eta} \frac{\pi(\eta)}{\pi(\tau)} \sum_{\nu \in X(k), \eta \subseteq \nu} \frac{\pi(\nu)}{\pi(\eta)} f(\nu) \\ &= \sum_{\nu \in X(k), \tau \subseteq \nu} \frac{\pi(\nu)}{\pi(\tau)} f(\nu) = \mathbb{E}_{\pi_\tau}[f_\tau]. \end{aligned}$$

◀

▶ **Lemma 32.** *Let X be a d-dimensional γ-local-spectral expander. Let $-1 \leq j \leq k \leq d$ and $f \in C^k(X)$. Then $f \in (\mathrm{Im}\, d_{j\nearrow k})^\perp$ if and only if f is from level $j+1$.*

Proof. First note that $f \in (\operatorname{Im} d_{j\nearrow k})^{\perp}$ if and only if $f \in \ker d^*_{j\nearrow k}$, if and only if $d^*_{j\nearrow k} f = 0$. Therefore, by Lemma 31, $\mathbb{E}_{\pi_\tau}[f_\tau] = 0$ for all $\tau \in X(j)$, which means f is from level $j+1$. ◀

Next, we provide a lemma that formulates the relation between functions that do not "come from below" and the subspaces U^j_k. But first, we need the following lemma.

▶ **Lemma 33.** *Let X be a d-dimensional γ-local-spectral expander. For all $-1 \le j \le k \le d$:*

$$C^k_0(X) \cap (d_{j\nearrow k} C^j_0(X))^{\perp} = (\operatorname{Im} d_{j\nearrow k})^{\perp}$$

Note that $\operatorname{Im} d_{j\nearrow k} := d_{j\nearrow k}(C^j(X)) = \{d_{j\nearrow k}(\psi) \mid \psi \in C^j(X)\}$ is the subspace of all k-cochains that come from j-cochain.

Proof. Since $(C^k_0(X) \cap (d_{j\nearrow k} C^j_0(X))^{\perp})^{\perp} = \operatorname{span}\{\mathbf{1}_k\} + d_{j\nearrow k} C^j_0(X)$, the lemma follows from the following equality: $d_{j\nearrow k}\mathbf{1}_j = \binom{k+1}{j+1}\mathbf{1}_k$. ◀

The following lemma shows that functions belong to $\bigoplus_{j<m\le k} U^m_k$ if and only if they are from level $j+1$.

▶ **Lemma 34.** *Let X be a d-dimensional γ-local-spectral expander. For all $-1 \le j \le k \le d$:*

$$\bigoplus_{j<m\le k} U^m_k = (\operatorname{Im} d_{j\nearrow k})^{\perp}$$

(we define the left-hand side to be the zero subspace when $j = k$).

Proof. The Lemma follows from Lemma 20, the definition of U^j_k, and Lemma 33. ◀

Now, let us study how well M^+_k shrinks functions that do not come from below. By the above lemmas, we already know that if a function does not come from below, it does not belong to U^j_k where j is low. Additionally, by the definition of W^j_k, we know that M^+_k shrinks $f \in W^j_k$ well when j is high. Therefore, our next step is to show how the two decompositions of $C^k_0(X)$: the spectral decomposition denoted W^j_k, and the combinatorial decomposition denoted U^j_k, are related.

From the work of [17] we already know that the projection of ϕ on W^j_k can be approximated by its projection on U^j_k:

▶ **Proposition 35** ([17, Theorem 5.10, simplified])**.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \le k < d$, and assume γ is small enough. For all $0 \le j \le k$ and $\phi \in C^k_0(X)$:*

$$(1 - b(k)\gamma)\|P_{U^j_k}\phi\| \le \|P_{W^j_k}\phi\| \le (1 + b(k)\gamma)\|P_{U^j_k}\phi\|$$

where $b(k)$ is an explicit positive constant such that $b(k)\gamma < 1$, and P is the orthogonal projection.

Let us prove that the two decompositions are actually identical:

▶ **Theorem 36** (Spectral decomposition and combinatorial decomposition are identical)**.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \le k < d$, and assume γ is small enough. Then $W^j_k = U^j_k$ for all $0 \le j \le k$.*

Proof. Let $0 \le j \le k$. We will prove that $U^j_k = W^j_k$ by showing that $(U^j_k)^{\perp} = (W^j_k)^{\perp}$. Let $\phi \in (U^j_k)^{\perp}$. Then $P_{U^j_k}\phi = 0$ and by Proposition 35 we get that $P_{W^j_k}\phi = 0$, and hence $\phi \in (W^j_k)^{\perp}$. Conversely, assume that $\phi \in (W^j_k)^{\perp}$, so $P_{W^j_k}\phi = 0$. By Proposition 35:

$$\frac{1}{1 + b(k)\gamma}\|P_{W^j_k}\phi\| \le \|P_{U^j_k}\phi\| \le \frac{1}{1 - b(k)\gamma}\|P_{W^j_k}\phi\|.$$

Hence $\|P_{U^j_k}\phi\| = 0$, and therefore $\phi \in (U^j_k)^{\perp}$. ◀

▶ **Corollary 37.** *Let X be a d-dimensional γ-local-spectral expander and assume γ is small enough. Let $-1 \leq j < k \leq d$ and $f \in C^k(X)$. Then f is from level $j+1$ if and only if $f \in W_k^{j+1} \oplus \cdots \oplus W_k^k$.*

Proof. The Corollary follows from Lemma 34, Lemma 32, and Theorem 36. ◀

Therefore, functions from "good neighborhoods" do not "come from below". By good neighborhoods we mean the subspaces W_k^j related to small eigenvalues of M_k^+.

We will now use Theorem 36 to improve some existing results. First, the following result improves the spectral decomposition of $C^k(X)$ for the case where X is a good enough expander. It easily follows from [17, Corollary 5.8]

▶ **Proposition 38.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \leq k < d$, and assume γ is small enough. Then for all $0 \leq j \leq k$:*

$$W_k^j = \operatorname{span}\left\{\phi \;\middle|\; M_k^+\phi = \mu\phi, \mu \in \left[\frac{k+1-j}{k+2} - \frac{\epsilon_k}{k+2}, \frac{k+1-j}{k+2} + \frac{\epsilon_k}{k+2}.\right]\right\}$$

Next, the following corollary improves [17, Theorem 5.9].

▶ **Corollary 39.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \leq k < d$, and assume γ is small enough. Then*

$$\operatorname{Spec}(M_k^+) \subseteq \{1\} \cup \bigcup_{j=0}^{k}\left[\frac{k+1-j}{k+2} - \frac{\epsilon_k}{k+2}, \frac{k+1-j}{k+2} + \frac{\epsilon_k}{k+2}\right].$$

Finally, the following corollary of Proposition 38 and Theorem 36 improves [17, Corollary 5.11].

▶ **Corollary 40.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \leq k < d$, and assume that γ is small enough. Then for all $i \in \mathbb{N}$ and $\phi \in C_0^k(X)$:*

$$\sum_{j=0}^{k}\left(\frac{k+1-j}{k+2} - \frac{\epsilon_k}{k+2}\right)^{2i}\left\|P_{U_k^j}\phi\right\|^2 \leq \|(M_k^+)^i\phi\|^2 \leq \sum_{j=0}^{k}\left(\frac{k+1-j}{k+2} + \frac{\epsilon_k}{k+2}\right)^{2i}\left\|P_{U_k^j}\phi\right\|^2$$

where P is the orthogonal projection.

Therefore if ϕ does not come from below then M_k^+ will shrink it better. For example, the best shrinking functions are from U_k^k.

Now, let us bound the shrinkage of f in a way that depends on its neighborhood.

▶ **Proposition 41.** *Let X be a d-dimensional γ-local-spectral expander, let $0 \leq k < d$, and assume γ is small enough. Let $f \in (\operatorname{Im} d_{j\nearrow k})^\perp$ for some $-1 \leq j < k$. Then*

$$\lambda_{M_k^+}(f) \leq \frac{k-j}{k+2} + \frac{\epsilon_k}{k+2}.$$

Proof. By Lemma 34, Theorem 36 and Proposition 38:

$$\begin{aligned} f \in (\operatorname{Im} d_{j\nearrow k})^\perp &= \bigoplus_{j<m\leq k} U_k^m = \bigoplus_{j<m\leq k} W_k^m \\ &= \operatorname{span}\left\{\phi \;\middle|\; M_k^+\phi = \mu\phi, \mu \in \left[\frac{1}{k+2} - \frac{\epsilon_k}{k+2}, \frac{k-j}{k+2} + \frac{\epsilon_k}{k+2}\right]\right\}. \end{aligned}$$

Lemma 28 finishes the proof. ◀

Next, our goal is to prove that there are many functions that shrink well. Therefore, we want to calculate the dimension of each W_k^j. First, let us prove the following lemmas.

▶ **Lemma 42.** *Let X be a d-dimensional γ-local-spectral expander and let $-1 \leq k < d$. If γ is small enough then d_k is one-to-one.*

Proof. Let μ be the smallest eigenvalue of M_k^+. By Corollary 39, $\mu \geq \frac{1-\epsilon_k}{k+2}$. Since γ is small enough, by Definition 18, $\epsilon_k < 1$ and therefore $\mu > 0$. Hence, M_k^+ is invertible. By Lemma 21, $d_k^* d_k = (k+2)M_k^+$ and therefore d_k is one-to-one. ◀

▶ **Lemma 43.** *Let X be a d-dimensional γ-local-spectral expander and let $-1 \leq j \leq k \leq d$. If γ is small enough then $\dim \operatorname{Im} d_{j\nearrow k} = \dim C^j(X)$ and $\dim(\operatorname{Im} d_{j\nearrow k})^\perp = \dim C^k(X) - \dim C^j(X)$.*

Proof. By Lemma 22, $d_{j\nearrow k} = \frac{1}{(k-j)!} d_{k-1} \dots d_j$. Therefore, since γ is small enough, by Lemma 42, $d_{j\nearrow k}$ is one-to-one. Hence, $\dim \ker d_{j\nearrow k} = 0$. By the rank-nullity theorem, $\dim \operatorname{Im} d_{j\nearrow k} + \dim \ker d_{j\nearrow k} = \dim C^j(X)$ and therefore $\dim \operatorname{Im} d_{j\nearrow k} = \dim C^j(X)$. Next, $(\operatorname{Im} d_{j\nearrow k})^\perp \oplus \operatorname{Im} d_{j\nearrow k} = C^k(X)$ implies that $\dim(\operatorname{Im} d_{j\nearrow k})^\perp = \dim C^k(X) - \dim \operatorname{Im} d_{j\nearrow k}$. Therefore, $\dim(\operatorname{Im} d_{j\nearrow k})^\perp = \dim C^k(X) - C^j(X)$. ◀

Since the combinatorial and spectral decompositions are equal, we can calculate the dimension of U_k^j instead of finding the dimension of W_k^j.

▶ **Proposition 44.** *Let X be a d-dimensional γ-local-spectral expander and let $0 \leq j \leq k \leq d$. If γ is small enough then $\dim U_k^j = |X(j)| - |X(j-1)|$.*

Proof. By [2, Proposition 2.43], $\dim(U_k^j \oplus \dots \oplus U_k^k) = \dim U_k^j + \dim(U_k^{j+1} \oplus \dots \oplus U_k^k)$. Applying Lemma 34, we get: $\dim(\operatorname{Im} d_{(j-1)\nearrow k})^\perp = \dim U_k^j + \dim(\operatorname{Im} d_{j\nearrow k})^\perp$. And by Lemma 43: $\dim C^k(X) - \dim C^{j-1}(X) = \dim U_k^j + \dim C^k(X) - \dim C^j(X)$. Therefore, $\dim U_k^j = \dim C^j(X) - \dim C^{j-1}(X) = |X(j)| - |X(j-1)|$. ◀

▶ **Corollary 45.** *Let X be a d-dimensional γ-local-spectral expander and let $0 \leq j \leq k < d$. If γ is small enough then $\dim W_k^j = |X(j)| - |X(j-1)|$.*

The above corollary shows that the amount of eigenvalues of M_k^+ in each neighborhood is not affected by the dimension k. We conclude that as j increases, W_k^j becomes much bigger, and hence most of the functions do not come from below.

Finally, we show how neighborhoods of different dimensions are connected.

▶ **Proposition 46.** *Let X be a d-dimensional γ-local-spectral expander and let $0 \leq j \leq k < d$. If γ is small enough then $d_k U_k^j = U_{k+1}^j$ and $d_k^* U_{k+1}^j = U_k^j$.*

The following corollary unravels what it means to "come from below" and provides a simpler decomposition of $C^k(X)$.

▶ **Corollary 47.** *Let X be a d-dimensional γ-local-spectral expander and let $0 \leq j \leq m \leq k < d$. If γ is small enough then:*

$$d_{m\nearrow k} W_m^j = d_{m\nearrow k} U_m^j = U_k^j = W_k^j \qquad \text{and} \qquad d^*_{m\nearrow k} W_k^j = d^*_{m\nearrow k} U_k^j = U_m^j = W_m^j.$$

In particular, for $m = j$, $W_k^j = d_{j\nearrow k} \ker d^*_{j-1}$ and therefore

$$C^k(X) = \operatorname{span}\{\mathbf{1}\} \oplus d_{0\nearrow k} \ker d^*_{-1} \oplus d_{1\nearrow k} \ker d_0^* \oplus \dots \oplus d_{k-1\nearrow k} \ker d^*_{k-2} \oplus \ker d^*_{k-1}.$$

6 Chernoff Bound for High-Dimensional Expanders

In this section, we apply our findings regarding high-order random walks and the Chernoff bound, to get a Chernoff bound for high-dimensional expander graphs. For functions that do not "come from below" and have a small variance, our bound is better than the naive HD Chernoff bound.

▶ **Theorem 48** (HD expander Chernoff bound). *Let X be a weighted d-dimensional γ-local-spectral expander, let $0 \le k < d$, and assume that γ is small enough. Let $\pi = \pi_k$ be the probability distribution over $X(k)$ and denote $\pi_* := \min_i \pi_i$. Let $w_1, \ldots, w_t$ be a random walk according to M_k^+ with initial distribution π. Let $f : X(k) \to \mathbb{R}$ such that f is from level $j+1$ (see Definition 1) for some $-1 \le j < k$. Then for all $\epsilon > 0$:*

$$\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \le \begin{cases} 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2}\right) & \mathrm{Var}_\pi(f) \le \frac{\epsilon^2\pi_*}{4h_k(j)^2} \\ 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2 \mathcal{L}^2}\right) & \textit{otherwise} \end{cases}$$

where $\mathcal{L} := \left\lceil \frac{\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*)}{1-h_k(j)} \right\rceil$, *and* $h_k(j) := \frac{k-j}{k+2} + \frac{\epsilon_k}{k+2}$ *where* $\lim_{\gamma\to 0}\epsilon_k = 0$ *(see Definition 18).*

Proof. First note that by Lemma 32, $f \in (\mathrm{Im}\, d_{j\nearrow k})^\perp$. By Lemma 34, $f \in C_0^k(X)$. By definition of $C_0^k(X)$, we know that $\mathbb{E}_\pi[f] = 0$. Hence, by Theorem 29, for every $\epsilon > 0$:

$$\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \le \begin{cases} 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2}\right) & \mathrm{Var}_\pi(f) \le \frac{\epsilon^2\pi_*}{4\lambda(f)^2} \\ 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2 \mathcal{Q}^2}\right) & \text{otherwise} \end{cases}$$

where $\mathcal{Q} := \left\lceil \frac{\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*)}{1-\lambda(f)} \right\rceil$. Now, by Proposition 41: $\lambda(f) \le \frac{k-j}{k+2} + \frac{\epsilon_k}{k+2} = h_k(j)$. Therefore $\mathcal{Q} \le \mathcal{L}$ and the theorem follows. ◀

▶ Remark 49. We can prove a more general version of this theorem, which applies to cases where f does come from below, but its mass on the lower subspaces is small. Denote $P_{k,j} := P_{(\mathrm{Im}\, d_{j\nearrow k})^\perp}$ the orthogonal projection on the upper subspaces. Denote $\ell := \|f - P_{k,j}f\|_\pi$ and assume $\ell < \epsilon\sqrt{\pi_*}/2$. Then for all $\epsilon > 0$:

$$\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \le \begin{cases} 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2}\right) & \mathrm{Var}_\pi(f) \le \left(\frac{\epsilon\sqrt{\pi_*} - 2\ell}{2h_k(j)}\right)^2 \\ 2\exp\left(-\frac{t\epsilon^2}{32\|f\|_\infty^2 \mathcal{L}^2}\right) & \text{otherwise} \end{cases}$$

where $\mathcal{L} := \left\lceil \frac{\ln(4\,\mathrm{Var}_\pi(f)/(\epsilon\sqrt{\pi_*} - 2\ell)^2)}{1-h_k(j)} \right\rceil$.

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 *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 Sheldon Jay Axler. *Linear algebra done right*, volume 2. Springer, third edition, 1997.

3 Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Math. J. (2)*, 19(3):357–367, 1967. doi:10.2748/tmj/1178243286.

4 Fan Chung and Linyuan Lu. Concentration inequalities and martingale inequalities: a survey. *Internet Mathematics*, 3(1):79–127, 2006.

5 Kai-Min Chung, Henry Lam, Zhenming Liu, and Michael Mitzenmacher. Chernoff-Hoeffding bounds for markov chains: Generalized and simplified. In *29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, February 29th - March 3rd, 2012, Paris, France*, pages 124–135, 2012. doi:10.4230/LIPIcs.STACS.2012.124.

6 Irit Dinur, Prahladh Harsha, Tali Kaufman, Inbal Livni Navon, and Amnon Ta Shma. List decoding with double samplers. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2134–2153. SIAM, 2019.

7 Irit Dinur and Tali Kaufman. High dimensional expanders imply agreement expanders. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 974–985, 2017. doi:10.1109/FOCS.2017.94.

8 Ankit Garg, Yin Tat Lee, Zhao Song, and Nikhil Srivastava. A matrix expander chernoff bound. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2018, pages 1102–1114, New York, NY, USA, 2018. ACM. doi:10.1145/3188745.3188890.

9 D Gillman. A chernoff bound for random walks on expander graphs. In *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, pages 680–691. IEEE, 1993.

10 David Gillman. A chernoff bound for random walks on expander graphs. *SIAM Journal on Computing*, 27(4):1203–1220, 1998.

11 Geoffrey Grimmett and David Stirzaker. *Probability and Random Processes*, pages 225, 227. Oxford University Press, third edition, 2001.

12 Alexander D Healy. Randomness-efficient sampling within nc. *Computational Complexity*, 17(1):3–37, 2008.

13 Shlomo Hoory, Nathan Linial, and Avi Wigderson. Expander graphs and their applications. *Bulletin of the American Mathematical Society*, 43(4):439–561, 2006.

14 Nabil Kahale. Large deviation bounds for markov chains. *Combinatorics, Probability and Computing*, 6(4):465–474, 1997.

15 Tali Kaufman and David Mass. High dimensional random walks and colorful expansion. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

16 Tali Kaufman and Izhar Oppenheim. High Order Random Walks: Beyond Spectral Gap. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2018)*, volume 116 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 47:1–47:17, 2018.

17 Tali Kaufman and Izhar Oppenheim. High order random walks: Beyond spectral gap. *Combinatorica*, pages 1–37, 2020.

18 Pascal Lezaud. Chernoff-type bound for finite markov chains. *Annals of Applied Probability*, pages 849–867, 1998.

19 Alexander Lubotzky. Expander graphs in pure and applied mathematics. *Bulletin of the American Mathematical Society*, 49(1):113–162, 2012.

20 Roy Wagner. Tail estimates for sums of variables sampled by a random walk. *Combinatorics, Probability and Computing*, 17(2):307–316, 2008.

21 Avi Wigderson and David Xiao. A randomness-efficient sampler for matrix-valued functions and applications. In *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 397–406. IEEE, 2005.

A Proving the Expander Chernoff Bound

Our proof is a generalization of [8]. As in [8], the proof is by reduction to the well-studied subject of martingales. First, let us provide the relevant definitions.

A.1 Preliminaries

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability-space: Ω is the sample space which is the set of possible outcomes of an experiment; $\mathcal{F}$ is a σ-algebra that is a set of events, where each event is a set of outcomes; and $\mathbb{P}$ is a probability measure function that assigns a probability to each event.

For example, in our case, the experiment is a random walk of length t on a graph $G = (V, E)$. A single outcome is a sequence of t vertices. The sample space is the set of possible sequences. The σ-algebra of our probability-space is the power set of the sample space. A set in this power set is an event (a set of walks), and the probability of an event is the probability that one of the walks in that event has occurred.

A probability-space can be equipped with a *filtration*, which is a monotone sequence of sub-σ-algebras $\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}$. Intuitively, each sub-$\sigma$-algebra $\mathcal{F}_i$ is the information known at time i (which includes what was known before that).

In our case, we are interested in the filtration generated by a random walk on the graph. Let $w_1, \ldots, w_t$ denote a random walk. Each w_i is a random variable from our sample space Ω to $V = \{1, \ldots, n\}$. The value of w_i given a sequence from Ω, is the i-th vertex in that sequence. The σ-algebra generated by $w_1, \ldots, w_i$ represents the information known after i steps and is denoted by $\sigma(w_1, \ldots, w_i)$. This σ-algebra is generated by the events $E_{v_1,\ldots,v_i}$ for all possible walks $(v_1, \ldots, v_i) \in V^i$, where $E_{v_1,\ldots,v_i}$ denotes the event of all possible outcomes that start with $v_1, \ldots, v_i$.

A sequence $\{X_n\}$ of random variables is said to be *adapted* to a filtration $\{\mathcal{F}_n\}$ if X_n is $\mathcal{F}_n$-measurable for all n. Informally, X_n is $\mathcal{F}_n$-*measurable* if X_n is known at time n. A random variable w_i is $\mathcal{F}_j$-measurable if $w_i^{-1}(v) \in \mathcal{F}_j$ for all $v \in V$, which means that w_i is known after j steps. Note that w_i is $\mathcal{F}_j$-measurable for $i \leq j$, and therefore $w_1, \ldots, w_t$ is adapted to the filtration $\{\sigma(w_1, \ldots, w_i)\}_{i=1}^t$.

We can now provide the definition of martingales and martingale difference sequences.

▶ **Definition 50** (Martingale, MDS). *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability-space and let $\{\mathcal{F}_n\}$ be a filtration. Let $\{X_n\}$ be a sequence of random variables such that $\{X_n\}$ is adapted to $\{\mathcal{F}_n\}$ and $\mathbb{E}[|X_n|] < \infty$ for all n.*

1. *If $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n] = X_n$ for all n then $\{X_n\}$ is a* martingale *with respect to $\{\mathcal{F}_n\}$.*
2. *If $\mathbb{E}[X_{n+1} \mid \mathcal{F}_n] = 0$ for all n then $\{X_n\}$ is an* MDS *(martingale difference sequence) with respect to the filtration $\{\mathcal{F}_n\}$.*

The following inequality is a generalization of the Chernoff bound to martingales.

▶ **Proposition 51** (Azuma's inequality [3]). *Let $X_0, \ldots, X_t$ be a martingale and suppose there is $c > 0$ such that $|X_i - X_{i-1}| \leq c$ for all $1 \leq i \leq t$ a.s. Then for all $\epsilon > 0$:*

$$\mathbb{P}(|X_t - X_0| \geq \epsilon) \leq 2 \exp\left(\frac{-\epsilon^2}{2tc^2}\right).$$

A.2 Reduction to Martingales

In this subsection we prove the following reduction from the problem of concentration for random walks to the problem of concentration for martingales. The proof is a generalization of the proof from [8, Theorem 1.6].

▶ **Proposition 52** (Reduction to martingales). *Let $M : l_2(\pi) \to l_2(\pi)$ be a connected non-bipartite random walk operator over state-space $V = \{1, \ldots, n\}$ with a stationary distribution π. Let $w_1, \ldots, w_t$ be a random walk according to M with initial distribution π and denote $\pi_* \coloneqq \min_i \pi_i$. Let $f \in l_2(\pi)$ be a function satisfying $\mathbb{E}_\pi[f] = 0$. Then for every $\epsilon > 0$, there is a martingale difference sequence $Z_1, \ldots, Z_t$ (see Definition 50) with respect to the filtration $\{\sigma(w_1, \ldots, w_i)\}_{i=1}^t$ such that $\frac{1}{t}\sum_{i=1}^t f(w_i) = W + \frac{1}{t}\sum_{i=1}^t Z_i$, where $|W| \leq \epsilon$ and each term Z_i satisfies $|Z_i| \leq 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(\|f\|_\pi / \epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}, 1 \right\} \right\rceil$.*

In order to prove this proposition, we first need to prove several lemmas. Let $K \in \mathbb{N}$. We will set it to its optimal value later in the subsection.

Our goal is to show that the sum $\sum_{i=1}^t f(w_i)$ is a martingale difference sequence with respect to the filtration $\{\sigma(w_1, \ldots, w_i)\}_{i=1}^t$, with a small error W. Ideally, we would have wanted $\mathbb{E}[f(w_{i+1}) \mid w_i]$ to be equal to 0. But in fact, the following holds:

▶ **Lemma 53.** *Let $1 \leq k \leq K$ and $1 \leq i \leq t-k$. Then $(M^k f)(w_i) = \mathbb{E}\left[(M^{k-1}f)(w_{i+1}) \;\middle|\; w_i\right]$.*

Proof. By definition of conditional expectation with respect to an event, we get that: $\mathbb{E}\left[(M^{k-1}f)(w_{i+1}) \;\middle|\; w_i\right] = \sum_{j=1}^n \mathbb{P}[w_{i+1} = j \mid w_i](M^{k-1}f)(j) = \sum_{j=1}^n M(w_i, j)(M^{k-1}f)(j)$ $= \big(M(M^{k-1}f)\big)(w_i) = (M^k f)(w_i)$. ◀

Therefore, to create an MDS out of $\sum_{i=1}^t f(w_i)$ we define the following random variables: $Y_1^{(k)} \coloneqq (M^{k-1}f)(w_1)$ for all $1 \leq k \leq K$, and $Y_i^{(k)} \coloneqq (M^{k-1}f)(w_i) - (M^k f)(w_{i-1})$ for all $1 < i \leq t - k + 1$.

▶ **Lemma 54.** *The following holds: $\sum_{i=1}^t f(w_i) = \sum_{k=1}^K \sum_{i=1}^{t-k+1} Y_i^{(k)} + \sum_{i=1}^{t-K} (M^K f)(w_i)$.*

Proof. Let $1 \leq m \leq t$ and $1 \leq k \leq K$. Then $\sum_{i=1}^m (M^{k-1}f)(w_i) = \sum_{i=1}^m Y_i^{(k)} + \sum_{i=1}^{m-1} (M^k f)(w_i)$, and the lemma is easy to prove by induction on K. ◀

The following corollary separates $\sum_{i=1}^t f(w_i)$ to an MDS part and an error part W. Later in this subsection, we will show that $\{Z_i\}$ is indeed an MDS and that W is small.

▶ **Corollary 55.** *Denote $W \coloneqq \frac{1}{t}\sum_{i=1}^{t-K} (M^K f)(w_i)$, and $Z_i \coloneqq \sum_{k=1}^{\min\{t-i+1, K\}} Y_i^{(k)}$ for all $1 \leq i \leq t$. Then $\frac{1}{t}\sum_{i=1}^t f(w_i) = W + \frac{1}{t}\sum_{i=1}^t Z_i$.*

Proof. By interchanging the order of summation,

$$\sum_{k=1}^K \sum_{i=1}^{t-k+1} Y_i^{(k)} = \sum_{i=1}^t \sum_{k=1}^{\min\{t-i+1, K\}} Y_i^{(k)} = \sum_{i=1}^t Z_i.$$

Lemma 54 finishes the proof. ◀

Our next step is to show that $\{Z_i\}$ is a martingale difference sequence with respect to the filtration $\{\sigma(w_1, \ldots, w_i)\}_{i=1}^t$. We prove it by showing that $\{Y_i\}$ is an MDS.

▶ **Lemma 56.** *For all* $1 \le k \le K$, $Y_1^{(k)}, \dots, Y_{t-k+1}^{(k)}$ *is an MDS with respect to the filtration* $\{\sigma(w_1, \dots, w_i)\}_{i=1}^{t-k+1}$.

Proof. First note that for all $0 \le k \le K$ and $1 \le i \le t-k$:

1. $(M^k f)(w_i)$ is $\sigma(w_1, \dots, w_i)$-measurable.
2. By the Markov property, $\mathbb{E}\big[(M^k f)(w_{i+1}) \bigm| w_1, \dots, w_i\big] = \mathbb{E}\big[(M^k f)(w_{i+1}) \bigm| w_i\big]$.

Let $1 \le k \le K$. By 1, $Y_i^{(k)}$ is $\sigma(w_1, \dots, w_i)$-measurable.

Secondly, we should prove that $Y_1^{(k)}, \dots, Y_{t-k+1}^{(k)}$ has the MDS property. Let $1 \le i \le t-k$. By 1, $(M^k f)(w_i)$ is $\sigma(w_1, \dots, w_i)$-measurable, hence by the stability property of conditional expectation:

$$\mathbb{E}\big[(M^k f)(w_i) \bigm| w_1, \dots, w_i\big] = (M^k f)(w_i). \tag{2}$$

Therefore, we have $\mathbb{E}\big[Y_{i+1}^{(k)} \bigm| w_1, \dots, w_i\big] = \mathbb{E}\big[(M^{k-1} f)(w_{i+1}) - (M^k f)(w_i) \bigm| w_1, \dots, w_i\big] = \mathbb{E}\big[(M^{k-1} f)(w_{i+1}) \bigm| w_i\big] - \mathbb{E}\big[(M^k f)(w_i) \bigm| w_1, \dots, w_i\big] = (M^k f)(w_i) - (M^k f)(w_i) = 0$ where the second equality is by 2 and the third equality is by Lemma 53 and (2). ◀

▶ **Corollary 57.** $Z_1, \dots, Z_t$ *is an MDS with respect to* $\{\sigma(w_1, \dots, w_i)\}_{i=1}^{t}$.

Now, let us bound the size of each Z_i. We need the following lemma:

▶ **Lemma 58.** *Let* M *be a connected random walk and let* $f \in l_2(\pi)$. *Then* $\|Mf\|_\infty \le \|f\|_\infty$.

Proof. Let $1 \le i \le n$. Then we have $|(Mf)(i)| \le \sum_j |M(i,j) f(j)| = \sum_j M(i,j)|f(j)| \le \sum_j M(i,j)\|f\|_\infty = \|f\|_\infty$, and therefore $\|Mf\|_\infty \le \|f\|_\infty$. ◀

▶ **Corollary 59.** *Let* $1 \le i \le t$. *Then* $|Z_i| \le 2K\|f\|_\infty$.

Proof. Since $|Z_i| = |\sum_{k=1}^{\min\{t-i+1,K\}} Y_i^{(k)}| \le \sum_{k=1}^{\min\{t-i+1,K\}} |Y_i^{(k)}| \le \sum_{k=1}^{K} |Y_i^{(k)}|$, we have

$$\begin{aligned} |Z_i| &\le \sum_{k=1}^{K} |(M^{k-1} f)(w_i) - (M^k f)(w_{i-1})| \le \sum_{k=1}^{K} \big(|(M^{k-1} f)(w_i)| + |(M^k f)(w_{i-1})|\big) \\ &\le \sum_{k=1}^{K} (\|M^{k-1} f\|_\infty + \|M^k f\|_\infty) \le \sum_{k=1}^{K} (\|f\|_\infty + \|f\|_\infty) = 2K\|f\|_\infty \end{aligned}$$

where the last inequality is by Lemma 58. ◀

Our last step before proving Proposition 52 is to bound the error $|W|$. First note that

▶ **Lemma 60.** *Let* $1 \le i \le n$. *Then* $\sqrt{\pi_i}|(M^K f)(i)| \le \lambda(f)^K \|f\|_\pi$.

Proof. By Lemma 27, we have that $\pi_i (M^K f)(i)^2 \le \sum_{j=1}^{n} \pi_j (M^K f)(j)^2 = \|M^K f\|_\pi^2 \le (\lambda(f)^K \|f\|_\pi)^2$ as needed. ◀

Let us now bound the error W.

▶ **Proposition 61.** $|W| \le \dfrac{\lambda(f)^K \|f\|_\pi}{\sqrt{\pi_*}}$.

Proof. From Lemma 60 we have:

$$|W| \leq \frac{1}{t}\sum_{i=1}^{t-K}|(M^K f)(w_i)| \leq \frac{1}{t}\sum_{i=1}^{t-K}\frac{1}{\sqrt{\pi_{w_i}}}\sqrt{\pi_{w_i}}|(M^K f)(w_i)|$$
$$\leq \frac{1}{t}\sum_{i=1}^{t-K}\frac{1}{\sqrt{\pi_{w_i}}}\cdot\lambda(f)^K\|f\|_\pi \leq \frac{t-K}{t}\lambda(f)^K\|f\|_\pi \max_i\frac{1}{\sqrt{\pi_i}} \leq \frac{\lambda(f)^K\|f\|_\pi}{\sqrt{\pi_*}}$$

where the last inequality is by definition of π_*. ◀

We can now prove Proposition 52.

Proof of Proposition 52. By Corollary 55: $\frac{1}{t}\sum_{i=1}^{t} f(w_i) = W + \frac{1}{t}\sum_{i=1}^{t} Z_i$ where

1. $W \coloneqq \frac{1}{t}\sum_{i=1}^{t-K}(M^K f)(w_i)$.
2. $Z_i \coloneqq \sum_{k=1}^{\min\{t-i+1,K\}} Y_i^{(k)}$ for all $1 \leq i \leq t$.

By Proposition 61, $|W| \leq \frac{\lambda(f)^K\|f\|_\pi}{\sqrt{\pi_*}}$. In order for $|W| \leq \epsilon$ to apply, we should choose K such that $\frac{\lambda(f)^K\|f\|_\pi}{\sqrt{\pi_*}} \leq \epsilon$. Note that by Proposition 8, $\lambda(f) < 1$. Therefore

$$K \geq \frac{\ln(\|f\|_\pi/\epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}. \tag{3}$$

By Corollary 57, $Z_1,\ldots,Z_t$ is an MDS with respect to $\{\sigma(w_1,\ldots,w_i)\}_{i=1}^t$. Moreover, by Corollary 59, $|Z_i| \leq 2K\|f\|_\infty$ for all $1 \leq i \leq t$. So to get the best bound on $|Z_i|$ we should choose the smallest K possible. Therefore, to comply with (3): $K \coloneqq \left\lceil \max\left\{\frac{\ln(\|f\|_\pi/\epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}, 1\right\}\right\rceil$ which yields the advertised bound on $|Z_i|$. ◀

▶ Remark 62. We can prove a more general version of this bound that applies to functions that most of their mass is on eigenspaces of small eigenvalues, and a small portion of their mass is on eigenspaces associated with large eigenvalues.

Let P be an orthogonal projection to a direct sum of some of the eigenspaces, such that $\lambda(Pf)$ is small and $\ell \coloneqq \|f - Pf\|_\pi$ is small too. Then $\|M^K f\|_\pi \leq \|M^K Pf\| + \|M^K(I-P)f\|_\pi \leq \lambda(Pf)^K\|Pf\|_\pi + \|(I-P)f\|_\pi \leq \lambda(Pf)^K\|f\|_\pi + \ell$. Therefore, in Lemma 60 we would get $\sqrt{\pi_i}|(M^K f)(i)| \leq \|M^K f\|_\pi \leq \lambda(Pf)^K\|f\|_\pi + \ell$. This means we would get that $|W| \leq \frac{\lambda(Pf)^K\|f\|_\pi + \ell}{\sqrt{\pi_*}}$ and therefore, for $\ell < \epsilon\sqrt{\pi_*}$ we get $|Z_i| \leq 2\|f\|_\infty\left\lceil \max\left\{\frac{\ln(\|f\|_\pi/(\epsilon\sqrt{\pi_*}-\ell))}{-\ln(\lambda(f))}, 1\right\}\right\rceil$.

A.3 Proof of Theorem 29

Proof. We prove an Expander Chernoff bound that depends on $\lambda(f)$. Let $\epsilon > 0$. Applying Proposition 52 with $\epsilon/2$ we get that there is an MDS $Z_1,\ldots,Z_t$ with respect to the filtration $\{\sigma(w_1,\ldots,w_i)\}_{i=1}^t$ such that $\frac{1}{t}\sum_{i=1}^t f(w_i) = W + \frac{1}{t}\sum_{i=1}^t Z_i$, where $|W| \leq \epsilon/2$ and each term

Z_i satisfies $|Z_i| \le 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(2\|f\|_\pi/\epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}, 1 \right\} \right\rceil$. In order to use Azuma's inequality, we first need to define a martingale out of our MDS $\{Z_i\}$. Let $X_0 = 0$ and $X_i \coloneqq X_{i-1} + Z_i$ for all $i \ge 1$. Then

$$|X_i - X_{i-1}| = |Z_i| \le 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(2\|f\|_\pi/\epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}, 1 \right\} \right\rceil, \tag{4}$$

$$\sum_{j=1}^{i} Z_j = \sum_{j=1}^{i} (X_j - X_{j-1}) = \sum_{j=1}^{i} X_j - \sum_{j=0}^{i-1} X_j = X_i - X_0. \tag{5}$$

▶ **Lemma 63.** *$X_1, \dots, X_t$ is a martingale with respect to $\{\sigma(w_1, \dots, w_i)\}_{i=1}^t$.*

Proof. First we'll prove that $X_1, \dots, X_t$ is adapted to the filtration, so we should prove that for all $i \ge 1$, X_i is $\sigma(w_1, \dots, w_i)$-measurable. Since $X_i = \sum_{j=1}^{i} Z_j$, it's enough to show that each Z_j is $\sigma(w_1, \dots, w_i)$-measurable when $j \le i$. From 57 we know that $Z_1, \dots, Z_t$ is an MDS with respect to our filtration, hence each Z_j is $\sigma(w_1, \dots, w_j)$-measurable. In addition, $\sigma(w_1, \dots, w_j) \subseteq \sigma(w_1, \dots, w_i)$ so each Z_j is also $\sigma(w_1, \dots, w_i)$-measurable. Therefore $X_1, \dots, X_t$ is adapted to the filtration.

Next we will prove that $X_1, \dots, X_t$ has the martingale property. Let $i \ge 1$. Since X_i is $\sigma(w_1, \dots, w_i)$-measurable, $\mathbb{E}[X_i \mid w_1, \dots, w_i] = X_i$ by the stability property of conditional expectation. In addition, since by Corollary 57 $Z_1, \dots, Z_t$ is an MDS, $\mathbb{E}[Z_{i+1} \mid w_1, \dots, w_i] = 0$. Therefore, $\mathbb{E}[X_{i+1} \mid w_1, \dots, w_i] = \mathbb{E}[X_i + Z_{i+1} \mid w_1, \dots, w_i] = \mathbb{E}[X_i \mid w_1, \dots, w_i] + \mathbb{E}[Z_{i+1} \mid w_1, \dots, w_i] = X_i$. ◀

Continuing with the proof of Theorem 29, by equation (5) we have that

$$\mathbb{P}\Big[\Big|\frac{1}{t}\sum_{i=1}^{t} f(w_i)\Big| > \epsilon\Big] = \mathbb{P}\Big[\Big|W + \frac{1}{t}\sum_{i=1}^{t} Z_i\Big| > \epsilon\Big] \le \mathbb{P}\Big[|W| + \Big|\frac{1}{t}\sum_{i=1}^{t} Z_i\Big| > \epsilon\Big]$$
$$\le \mathbb{P}\Big[\Big|\frac{1}{t}\sum_{i=1}^{t} Z_i\Big| > \frac{\epsilon}{2}\Big] = \mathbb{P}\Big[|\sum_{i=1}^{t} Z_i| > \frac{t\epsilon}{2}\Big] = \mathbb{P}\Big[|X_t - X_0| > \frac{t\epsilon}{2}\Big],$$

Let $c \coloneqq 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(2\|f\|_\pi/\epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}, 1 \right\} \right\rceil$. By (4) we know that $|X_i - X_{i-1}| \le c$. Furthermore, by Lemma 63, X_t is a martingale. Therefore we can apply Azuma's inequality to get: $\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \le 2\exp\left(\frac{-(\frac{t\epsilon}{2})^2}{2tc^2}\right) = 2\exp\left(-\frac{t^2\epsilon^2}{8tc^2}\right) = 2\exp\left(-\frac{t\epsilon^2}{8c^2}\right)$.

Since $\mathbb{E}_\pi[f] = 0$, we get $\mathrm{Var}_\pi(f) \coloneqq \sum_i \pi_i (f(i) - \mathbb{E}_\pi[f])^2 = \sum_i \pi_i f(i)^2 = \|f\|_\pi^2$, so

$$c = 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(2\sqrt{\mathrm{Var}_\pi(f)}/\epsilon\sqrt{\pi_*})}{-\ln(\lambda(f))}, 1 \right\} \right\rceil = 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*)}{-2\ln(\lambda(f))}, 1 \right\} \right\rceil.$$

Note that $c = 2\|f\|_\infty$ if $\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*) \le -2\ln(\lambda(f))$, or $\mathrm{Var}_\pi(f) \le \epsilon^2\pi_*/4\lambda(f)^2$. Moreover, $c \le 2\|f\|_\infty \left\lceil \max\left\{ \frac{\ln(4\,\mathrm{Var}_\pi(f)/\epsilon^2\pi_*)}{1-\lambda(f)}, 1 \right\} \right\rceil$ since $2\ln(r) \le r - 1$ for all $0 < r \le 1$. Hence,

$$\mathbb{P}[|\frac{1}{t}\sum_{i=1}^{t} f(w_i)| > \epsilon] \le \begin{cases} 2\exp\Big(-\frac{t\epsilon^2}{32\|f\|_\infty^2}\Big) & \mathrm{Var}_\pi(f) \le \frac{\epsilon^2\pi_*}{4\lambda(f)^2} \\ 2\exp\Big(-\frac{t\epsilon^2}{32\|f\|_\infty^2\mathcal{L}^2}\Big) & \text{otherwise.} \end{cases}$$ ◀

Vector-Matrix-Vector Queries for Solving Linear Algebra, Statistics, and Graph Problems

Cyrus Rashtchian
Department of Computer Science & Engineering, UC San Diego, CA, USA
crashtchian@eng.ucsd.edu

David P. Woodruff
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@cs.cmu.edu

Hanlin Zhu
Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing, China
zhuhl17@mails.tsinghua.edu.cn

Abstract

We consider the general problem of learning about a matrix through vector-matrix-vector queries. These queries provide the value of $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ over a fixed field $\mathbb{F}$ for a specified pair of vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}^n$. To motivate these queries, we observe that they generalize many previously studied models, such as independent set queries, cut queries, and standard graph queries. They also specialize the recently studied matrix-vector query model. Our work is exploratory and broad, and we provide new upper and lower bounds for a wide variety of problems, spanning linear algebra, statistics, and graphs. Many of our results are nearly tight, and we use diverse techniques from linear algebra, randomized algorithms, and communication complexity.

2012 ACM Subject Classification Theory of computation → Stochastic approximation

Keywords and phrases Query complexity, property testing, vector-matrix-vector, linear algebra, statistics, graph parameter estimation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.26

Category RANDOM

Funding *David P. Woodruff*: D. Woodruff would like to thank support in part by the Office of Naval Research (ONR) grant N00014-18-1-2562.

1 Introduction

In the past few decades, there has been a significant amount of research on query-based algorithms, motivated by compressed sensing, streaming, sketching, distributed methods, graph parameter estimation, and property testing [14, 23, 26, 43, 45]. Most of this work focuses on local queries that only access a small portion of the unknown data at a time. For example, prior work on graph parameter estimation has considered *degree* queries (which output the degree of a vertex v), *edge existence* queries (which answer whether a pair $\{u, v\}$ forms an edge), and *neighbor* queries (which provide the i^{th} neighbor of a vertex v). Not surprisingly, such queries have limited utility for certain problems. Even estimating the number of edges in a graph is known to require a polynomial number of edge existence, degree, and neighbor queries [24, 25].

This has led researchers to consider queries that still reveal a small amount of information, while being more global in nature. For example, *bipartite independent set* queries (which indicate whether or not there is at least one edge between two disjoint sets of vertices) can be used to estimate the number of edges with only polylog(n) queries [9, 19]. Similarly, *cut* queries (which provide the number of edges crossing a graph cut) can be used to find the


© Cyrus Rashtchian, David P. Woodruff, and Hanlin Zhu;
licensed under Creative Commons License CC-BY



exact minimum cut in a graph [39, 35]. Augmenting edge existence, degree, and neighbor queries with access to an edge sampling oracle (which provides a uniformly random edge) leads to elegant algorithms for estimating the number of certain subgraphs (e.g., triangles or cliques) [5], which was a major open problem (without edge sampling) until recently [21, 40].

As the diversity of queries increases (along with the range of applicable problems), it is natural to wonder whether there is a more general framework for understanding the power and limitations of query-based algorithms. In this work, we initiate the study of querying a matrix through bilinear forms, which generalizes the above mentioned queries and several more (sometimes with an $O(\log n)$ factor overhead). Formally, let $\boldsymbol{M}$ be an $n \times n$ matrix over a field $\mathbb{F}$. We consider vector-matrix-vector queries, which we call $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries for short. Given a pair of vectors $\boldsymbol{u}, \boldsymbol{v} \in \mathbb{F}^n$, these queries return the value of $\boldsymbol{u}^\mathsf{T}\boldsymbol{M}\boldsymbol{v}$ over $\mathbb{F}$. For graph applications, we often let the matrix $\boldsymbol{M}$ be the adjacency matrix of a graph. We later explain how to simulate standard graph queries with $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Allowing $\boldsymbol{M}$ to take values in other fields enables us to consider a greater variety of linear algebra, statistics, and data analytic problems. The underlying field $\mathbb{F}$ will play an important role in our results, where working over $\mathbb{F}_2$ or $\mathbb{R}$ will change the query complexity of certain problems. We assume that the entries have $O(\log n)$ bit-complexity, and therefore, the output of one $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query provides only $O(\log n)$ bits of information. We strive for algorithms using a subquadratic number of queries, which allows us to solve the problem without trivially learning the whole matrix. Unless we specify otherwise, we allow the queries to be randomized and adaptive.

From a practical point of view, algorithms based on $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries would most likely be useful in the context of specialized hardware or distributed environments. Computing a query only requires a weighted sum of entries of $\boldsymbol{M}$, and hence, it would be easy to execute in a massively parallel fashion. For example, if each processor handled a single row, then the local memory would be bounded by $O(n \log n)$ for storing $\boldsymbol{u}$ and $\boldsymbol{v}$. In a shared-nothing system, the number of communication rounds would be proportional to the number of queries. Similarly, in a streaming environment where single entries of $\boldsymbol{M}$ are changed at each step, the memory would be $O(\log n)$ times the number of queries. Working over a finite field $\mathbb{F}$ would reduce the memory overhead to $O(\log |\mathbb{F}|)$.

That being said, our focus is on the theoretical aspects of the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query model. We consider many problems, spanning linear algebra, statistics, and graph properties. Part of our motivation comes from finding algorithms that are query-efficient in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model, while surpassing lower bounds for more restricted models. For example, we consider properties that depend on the whole matrix (e.g., having low rank, being unitary or doubly stochastic) or the entire graph (e.g., being a perfect matching or a star). As these are global properties, it is intuitively challenging to verify them using local queries without simply learning the whole matrix or graph. Overall, the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query model opens up many theoretical directions, and it facilitates new connections between linear algebra, randomized algorithms, and communication complexity.

1.1 Related work and other queries

The $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model provides a unifying lens and generalizes many previously studied queries. We give a brief overview of how to simulate other models and mention other related work.

- **Standard Graph Queries.** To gain intuition about $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries, we note that if $\boldsymbol{M}$ is the adjacency matrix of a graph, then a single query over a large field (e.g., $\mathbb{Q}$ or $\mathbb{R}$) provides the exact edge count. It is also easy to show that $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries suffice to simulate degree, edge existence, neighbor, or edge sampling queries

(see Section A.1 for details). Therefore, $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries achieve a variety of previous results with only an $O(\log n)$ factor overhead, such as estimating the number of cliques of different sizes [2, 5, 21, 22, 40], the number of stars [27], and the minimum vertex cover [37].

- **Independent Set Queries.** Another line of work considers *independent set* oracles for graphs (which return whether a given set of vertices induces an independent set or contains at least one edge), mostly in the context of estimating the number of edges in a graph [9, 17, 18, 19]. Interestingly, bipartite independent set queries are known to be much stronger than independent set queries [9, 17]. A special case of bipartite independent set query (where one of the bipartition sets is a singleton) has been used for testing k-colorability of graphs [10], and high degree vertex discovery [44]. While these algorithms are randomized and approximate, prior work also studies exact graph learning problems [1, 3, 4]. When $\boldsymbol{M}$ is a binary matrix over a large enough field (e.g., $\mathbb{Q}$ or $\mathbb{R}$), then $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries generalize both independent and bipartite independent set queries by taking $\boldsymbol{u}$ and $\boldsymbol{v}$ to be indicator vectors for the sets. In particular, the power of the bipartite version motivates allowing $\boldsymbol{u}$ and $\boldsymbol{v}$ to differ in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model.
- **Fine-Grained Complexity.** Independent set queries are partially motivated by studying the complexity of decision vs. counting problems [18, 19]. While we do not know of a (natural) use of $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries in this area, future work could consider using our algorithms for a similar complexity-theoretic reduction. Our model could also be extended to tensors, where queries are k-linear forms, analogous the generalization to k-partite independent set queries for counting k-cliques, which has applications to k-SUM and related problems [19].
- **Cut Queries.** Another global graph query model considers *cut* queries (which provide the number of edges in a graph $G = (V, E)$ crossing a cut $(S, V \setminus S)$). It is known that $\widetilde{O}(n)$ cut queries suffice to exactly compute a minimum cut in a graph, and $\widetilde{O}(n^{5/3})$ suffice to compute an s-t cut [39]. This has also been extended to multigraphs [35]. We can directly simulate cut queries via indicator vectors $\boldsymbol{u} = \mathbf{1}_S$ and $\boldsymbol{v} = \mathbf{1}_{\{V\setminus S\}}$, when $\boldsymbol{M}$ is the adjacency matrix of the graph. As the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model is more general than cut queries, it an interesting open question whether a sublinear number of queries suffice for these problems.
- **Matrix-Vector Queries.** A similar but more powerful query model considered by previous work involves matrix-vector queries [42]. In this case, the queries return a vector of n values $\boldsymbol{v}^\mathrm{T}\boldsymbol{M}$ or $\boldsymbol{M}\boldsymbol{v}$ when given a vector $\boldsymbol{v} \in \mathbb{F}^n$. We study many of the same problems as this prior work. In some cases, we show that certain problems (such as determining if a matrix is symmetric or diagonal) have constant query complexity in both models, even though $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries reveal much less information than matrix-vector queries. Previous work also considers lower bounds for the operator norm in the matrix-vector model [13]. There is also work on the query complexity of computing PCA in this model [41]. Finally, we provide examples where matrix-vector queries are more powerful because there are lower bounds for $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries (see, e.g., Section 3.1).
- $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ **Data Structures.** A complementary line of work considers the data structure complexity of $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries [15, 16, 20, 30, 36]. More precisely, the goal is to preprocess $\boldsymbol{M}$ using a small amount space so that the value of $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v}$ can be obtained with a small query time (e.g., in the cell-probe model or natural restrictions of that model). Since there are connections between such data structures and challenging complexity theoretic problems (e.g., matrix rigidity, see [20, 36]), it is an outstanding question to further explore whether our results have implications for $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ data structures or vice versa.

Table 1 Our upper and lower bounds on the query complexity in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model for $n \times n$ matrices and constant success probability. Results hold over any field unless stated otherwise.

Linear Algebra Problems		
Schatten p-norm	$\Omega(\sqrt{n})$ for $p \in [0, 4)$, const. factor approx. over $\mathbb{R}$	Theorem 2
	$\Omega(n^{1-2/p})$ for $p \geq 4$, const. factor approx. over $\mathbb{R}$	Theorem 2
Rank testing	$\Omega(k^2)$ to distinguish rank k vs. $k+1$ over $\mathbb{F}_p$	Theorem 3
	$\Omega(n^{2-O(\varepsilon)})$ for $(1 \pm \epsilon)$ approx. over $\mathbb{R}$, non-adaptive	Theorem 4
Trace estimation	$\Omega(n/\log n)$ and $O(n)$ for entries in $\{0, 1, 2, \ldots, n^3\}$	Theorem 5
Diagonal matrix	$O(1)$	Theorem 6
Symmetric matrix	$O(1)$	Theorem 7
Unitary matrix	$\Omega(n/\log n)$ and $O(n)$ for randomized queries over $\mathbb{C}$	Theorem 8
	$\Omega(n^2/\log n)$ for deterministic queries over $\mathbb{C}$	Theorem 10
Statistics Problems		
All ones column	$\Omega(n/\log n)$ and $O(n)$ over $\mathbb{R}$	Section 4.1
Two identical columns	$\Omega(n)$ and $O(n \log n)$ over $\mathbb{F}_2$	Section 4.2
	$O(n)$ over $\mathbb{R}$	Section 4.2
Column-wise majority	$\Theta(n^2)$ over $\mathbb{F}_2$	Corollary 15
Permutation matrix	$O(1)$ over $\mathbb{R}$	Theorem 16
	$\Omega(n)$ over $\mathbb{F}_2$	Theorem 17
Doubly stochastic matrix	$O(1)$ over $\mathbb{R}$	Theorem 18
Negative entry detection	$\Omega(n^2/\log n)$ over $\mathbb{R}$	Theorem 19
Graph Problems		
Triangle detection	$\Omega(n^2/\log n)$	Theorem 20
Star graph	$O(1)$ over $\mathbb{R}$	Theorem 21
Local graph queries	$O(\log n)$	Lemma 22

1.2 Our Results

We provide new upper and lower bounds on the query complexity of various problems in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model. Table 1 summarizes our results. Many of the bounds are nearly tight: for some problems $O(1)$ queries suffice, and for others, either $\widetilde{\Theta}(n)$ or $\widetilde{\Theta}(n^2)$ are necessary and sufficient. For brevity, we defer formal definitions to the relevant subsections. Here we highlight some interesting results.

Linear Algebra Problems. Section 3.1 provides lower bounds for approximately computing many matrix norms, such as the trace norm, Frobenius norm, and operator norm (in general, we study Schatten p-norms; see Section 3.1 for the definition). To prove this result, we develop a general simulation result that allows us to establish lower bounds for adaptive $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries by reducing them to lower bounds for non-adaptive *entry-wise* queries. The key idea is that such a simulation result holds whenever the input matrix distribution is rotationally invariant (under row permutations). Then, we utilize known sketching lower bounds for matrix norms that identify a hard distribution that is rotationally invariant [31].

We give constant-query algorithms for testing if a matrix is diagonal (Section 3.4) or symmetric (Section 3.5). While these algorithms are fairly straightforward, they exhibit the power of $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries to efficiently test for global properties of the matrix. We prove nearly-matching bounds for testing if a matrix is orthonormal (over $\mathbb{R}$) or unitary (over $\mathbb{C}$). The lower bound uses an encoding of information via the Hadamard matrix.

Statistics Problems. Turning to other matrix problems, we consider properties of one or more columns (our results also hold for rows, by symmetry of the query model). For example, Section 4.1 and Section 4.2 provide nearly matching upper and lower bounds for testing if there is an all ones column or two identical columns. Many of our lower bounds follow from communication complexity via Disjointness. While this may be technically simple, we note that our reductions require certain gadgets that seem to be new in the context of query complexity; for example, see our lower bounds for permutation matrices (Theorem 17). This also led us to study negative entry detection in its own right, because a lower bound of $\Omega(n^2/\log n)$ from Theorem 19 essentially provides the reason why certain results for binary matrices (e.g., graphs) cannot be generalized.

Graph Problems. Our upper bound on permutation matrices (Theorem 16) gives a constant-query algorithm for detecting whether a graph is a perfect matching. We also provide a constant-query upper bound for testing whether a graph is a star on n vertices (Theorem 21). Both of these are global properties that would be difficult to verify using standard graph queries. They also complement previous results for learning hidden matchings or other structures using independent set queries [3, 4]. As mentioned previously, being able to simulate local graph queries with $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries gives rise to a number of results on graph parameter estimation in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model (see, e.g., [2, 5, 21, 22, 27, 40, 37]).

Organization. We start with preliminaries in Section 2. We provide results for linear algebra problems in Section 3, for statistics problems in Section 4, and for graph problems in Section 5. We conclude in Section 6.

2 Preliminaries

We use capital bold letters $(\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{M}, \ldots)$ to represent matrices, lower-case bold letters $(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{x}, \boldsymbol{y}, \ldots)$ to represent column vectors. We use non-bold lower-case letters $(x, y, \ldots)$ to represent strings. For a matrix $\boldsymbol{M}$, we let M_{ij} denote the entry in i^{th} row and j^{th} column. For a vector $\boldsymbol{v}$, we use v_i to denote the i^{th} entry. For a string x, we use x_i to denote the i^{th} entry. We use $\mathbb{F}$ to represent arbitrary fields, and use $\mathbb{F}_p$ to represent the finite field with p elements where p is prime, and $\mathbb{R}$ to denote the reals. We use $G = (V, E)$ to represent a simple graph, where V denotes the set of vertices and E denotes the set of edges. We query the adjacency matrix. Some of our lower bounds use the communication complexity of Disjointness, where Alice has $x \in \{0,1\}^n$, Bob has $y \in \{0,1\}^n$, and they decide if there exists an index i with $x_i = y_i = 1$. The randomized communication complexity is $\Omega(n)$ [29, 38]. We also use the following result: if x and y contain exactly $n/4$ ones, then the randomized complexity is still $\Omega(n)$ [7, 28].

3 Linear Algebra Problems

3.1 Lower Bounds for Approximating Matrix Norms

Say that a distribution over matrices $\boldsymbol{X} \in \mathbb{R}^{n \times n}$ is *orthonormal and rotationally invariant* if all rows of each $\boldsymbol{X}$ in the support are orthonormal and the distribution remains the same under any permutation of the rows of $\boldsymbol{X}$. We will consider distributions over matrices $\boldsymbol{M}$ formed by fixing a diagonal matrix $\boldsymbol{\Sigma}$, sampling two matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$ from orthonormal and rotationally invariant distributions, and letting $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{Y}^{\mathrm{T}}$. At a high level, are interested in algorithms for computing functions of the singular values $\boldsymbol{\Sigma}$, which remain invariant over matrices in such distributions.

Our first goal is to prove a structural result relating $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries to entry-wise queries of $\boldsymbol{M}$. Then, we use this reduction to prove new lower bounds. To do so, we utilize known streaming lower bounds, and we take advantage of the fact that these lower bounds are based on hard distributions that are orthonormal and rotationally invariant. Recall that $[s] = \{1, 2, \ldots, s\}$ and that $\boldsymbol{e}_i \in \{0,1\}^n$ denotes the i^{th} standard basis vector.

▶ **Lemma 1.** *Let $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{Y}^\mathrm{T}$ be a random $n \times n$ real-valued matrix, where $\boldsymbol{\Sigma}$ is diagonal, and $\boldsymbol{X}$ and $\boldsymbol{Y}$ are sampled from orthonormal and rotationally invariant distributions. Any $s \leq n$ deterministic, adaptive queries in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model can be simulated by s^2 non-adaptive entry-wise queries to the values of $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_j$ for $i, j \in [s]$.*

Proof. We proceed by induction on the number of queries $s \geq 1$. For the base case, consider a query $\boldsymbol{u}_1\boldsymbol{M}\boldsymbol{v}_1$, where $\boldsymbol{u}_1, \boldsymbol{v}_1$ are arbitrary unit vectors. Observe that $\boldsymbol{u}_1^\mathrm{T}\boldsymbol{X}$ and $\boldsymbol{Y}^\mathrm{T}\boldsymbol{v}_1$ are random unit vectors, and moreover, they follow the same distribution as $\boldsymbol{e}_1^\mathrm{T}\boldsymbol{X}$ and $\boldsymbol{Y}^\mathrm{T}\boldsymbol{e}_1$, respectively. Since $\boldsymbol{M} = \boldsymbol{X}\boldsymbol{\Sigma}\boldsymbol{Y}^\mathrm{T}$, we see that the values of $\boldsymbol{u}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_1$ and $\boldsymbol{e}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_1$ are identically distributed as well.

Suppose the lemma holds for any $s-1$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Consider a sequence of s queries

$$\boldsymbol{u}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_1,\ \boldsymbol{u}_2^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_2,\ \ldots\ ,\ \boldsymbol{u}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{v}_s, \tag{1}$$

for unit vectors $\boldsymbol{u}_i, \boldsymbol{v}_i$ for $i \in [s]$ that may depend adaptively on the previous queries. Assume without loss of generality that $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_s$ and $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_s$ are respectively linearly independent. For the final query vectors $\boldsymbol{u}_s$ and $\boldsymbol{v}_s$, decompose them as

$$\boldsymbol{u}_s = \boldsymbol{a}_s + \boldsymbol{b}_s \qquad \text{and} \qquad \boldsymbol{v}_s = \boldsymbol{c}_s + \boldsymbol{d}_s,$$

where

$$\boldsymbol{a}_s \in \mathsf{span}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{s-1}\} \qquad \text{and} \qquad \boldsymbol{c}_s \in \mathsf{span}\{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_{s-1}\},$$

and where $\boldsymbol{a}_s$ is orthogonal to $\boldsymbol{b}_s$, and $\boldsymbol{c}_s$ is orthogonal to $\boldsymbol{d}_s$.

Invoking the inductive hypothesis, this decomposition implies that $\boldsymbol{a}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{c}_s$ can be simulated using $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_j$ for $i, j \in [s-1]$. Furthermore, by the orthogonality assumptions, we have that $\boldsymbol{b}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s$ follows the same distribution as $\boldsymbol{e}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$, even conditioned on the previous queries.

It remains to argue about $\boldsymbol{a}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s$ and $\boldsymbol{b}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{c}_s$. We begin with the former, noting that the latter follows by a symmetric argument. Let $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_{s-1}$ denote an orthonormal basis for $\mathsf{span}\{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_{s-1}\}$. Considering the expansion of $\boldsymbol{a}_s$ in this basis, we observe that, by linearity, it suffices to simulate

$$\boldsymbol{w}_1^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s,\ \boldsymbol{w}_2^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s,\ \ldots\ ,\ \boldsymbol{w}_{s-1}^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s \tag{2}$$

using only the information from $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$ for $i \in [s-1]$. To establish this, consider any vector $\boldsymbol{w}_i$ for $i \in [s-1]$. By assumption, $\boldsymbol{X}$ and $\boldsymbol{Y}$ are drawn from orthonormal and rotationally invariant distributions. Since $\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_s$ form an orthonormal basis, we have that $\boldsymbol{w}_i^\mathrm{T}\boldsymbol{X}$ is a random unit vector following the same distribution as $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{X}$. Moreover, by orthogonality, for any $i \geq 2$, the distribution of $\boldsymbol{w}_i^\mathrm{T}\boldsymbol{X}$ remains the same as $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{X}$ even conditioned on

$$\boldsymbol{w}_1^\mathrm{T}\boldsymbol{X},\ \boldsymbol{w}_2^\mathrm{T}\boldsymbol{X},\ \ldots\ ,\ \boldsymbol{w}_{i-1}^\mathrm{T}\boldsymbol{X}.$$

an analogous argument implies that $\boldsymbol{Y}^\mathrm{T}\boldsymbol{d}_s$ follows the same distribution as $\boldsymbol{Y}^\mathrm{T}\boldsymbol{e}_s$, even conditioned on the previous queries. Therefore, we have that $\boldsymbol{w}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{d}_s$ is identically distributed as $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$. As this holds for all $i \in [s-1]$, the queries in Eq. (2) can be simulated by $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_s$ for $i \in [s-1]$. By symmetry, a similar result holds for simulating $\boldsymbol{b}_s^\mathrm{T}\boldsymbol{M}\boldsymbol{c}_s$. Therefore, we have shown that all s deterministic queries in Eq. (1) can be simulated by the s^2 entry-wise non-adaptive queries to $\boldsymbol{e}_i^\mathrm{T}\boldsymbol{M}\boldsymbol{e}_j$ for $i, j \in [s]$, as desired. ◀

We use this structural result to prove lower bounds for computing certain matrix norms by applying sketching lower bounds due to Li, Nguyen, and Woodruff [31]. For $p \in (0, \infty)$, the *Schatten p-norm* of a real matrix $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ with singular values $\sigma_1, \ldots, \sigma_n$ is defined as

$$\|\boldsymbol{M}\|_p = \left(\sum_{i=1}^{n} \sigma_i^p \right)^{1/p}.$$

By convention, the Schatten 0-norm is the rank of the matrix, and the Schatten ∞-norm equals the largest singular value (a.k.a., operator norm). We have the following result for the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model.

▶ **Theorem 2.** *Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be a matrix. For any value $p \in [0, 4)$, computing a constant-factor approximation to the Schatten p-norm of $\boldsymbol{M}$ requires $\Omega(\sqrt{n})$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. For $p \geq 4$, computing a constant-factor approximation to the Schatten p-norm of $\boldsymbol{M}$ requires $\Omega(n^{1-2/p})$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Both results hold for randomized, adaptive queries with constant success probability.*

We sketch the proof of this theorem, which now follows directly from previous results. Before applying Lemma 1, we use Yao's principle [46] to show that it suffices to consider deterministic query algorithms for distributions over input matrices. Also, the query vectors can be taken to be unit vectors without loss of generality, as the algorithm can rescale the results. Then, we note that the previous lower bounds use hard distributions that are orthonormal and rotationally invariant [31]. As a result, the distribution of matrices $\boldsymbol{M}$ satisfies the conditions of Lemma 1.

The previous results hold over the *bilinear sketching model*, where the sketches correspond to an $r \times n$ matrix $\boldsymbol{U}$ and an $s \times n$ matrix $\boldsymbol{V}$, and the goal is to approximate $\|\boldsymbol{M}\|_p$ up to a constant factor using $\boldsymbol{U}\boldsymbol{M}\boldsymbol{V}^\mathrm{T}$. Applying Lemma 1, we see that any algorithm making s queries in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model corresponds to a bilinear sketch with both matrices being $s \times n$. Moreover, as the conclusion of the lemma only uses entry-wise queries, the corresponding matrices consist of the $s \times s$ identity matrix in the upper left-hand corner, while the rest of the matrix is all zeroes. The lower bound on bilinear sketches implies

- $s^2 = \Omega(n)$ for approximating the Schatten p-norm with $p \in [0, 4)$
- $s^2 = \Omega(n^{2-4/p})$ for approximating the Schatten p-norm with $p \geq 4$.

Taking a square root leads to the bounds in Theorem 2.

The above provides separations between the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ and matrix-vector models [42]. Indeed, it is known that there exist non-trivial bilinear sketching matrices for approximating the Schatten p-norm whenever p is an even integer. Denoting such sketching matrices as $\boldsymbol{U}$ and $\boldsymbol{V}$, it suffices for $\boldsymbol{U}$ and $\boldsymbol{V}^\mathrm{T}$ to each have $O(n^{1-2/p})$ rows [31] to approximate the Schatten p-norm up to a constant factor. Observe that the Schatten p-norm of a matrix $\boldsymbol{M}$ is the same as the Schatten $p/2$-norm of the matrix $\boldsymbol{M}\boldsymbol{M}^\mathrm{T}$. Thus, if p is an integer multiple of 4, then in the matrix-vector model one can first compute $\boldsymbol{U}\boldsymbol{M}$ and then compute $\boldsymbol{M}^\mathrm{T}\boldsymbol{V}$, and then multiply these together to obtain $\boldsymbol{U}\boldsymbol{M}\boldsymbol{M}^\mathrm{T}\boldsymbol{V}$, where $\boldsymbol{U}$ and $\boldsymbol{V}$ are the corresponding sketching matrices for the Schatten $p/2$-norm. The total cost is $O(n^{1-4/p})$ queries in the matrix-vector model.

On the other hand, Theorem 2 implies that $\Omega(n^{1-2/p})$ queries are necessary in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model, thus providing a separation for integers $p \geq 4$ which are multiples of 4. We also directly get an $\Omega(n)$ lower bound for approximating the operator norm up to a constant factor, using the $\Omega(n^2)$ lower bound bound for general sketches in [33]. For recent work on actually finding the top eigenvector and solving a linear system in the matrix-vector model in the high accuracy regime, see [13].

3.2 Rank Testing

Given a matrix $\boldsymbol{M} \in \mathbb{F}^{n\times n}$, a natural problem is to determine the rank of $\boldsymbol{M}$. We first consider matrices over a finite field $\mathbb{F}_p$ for a prime p.

▶ **Theorem 3.** *Given a matrix $\boldsymbol{M} \in \mathbb{F}_p^{n\times n}$ and an integer k, at least $\Omega(k^2)$ adaptive queries are necessary to decide the rank whether the rank of $\boldsymbol{M}$ is k or $k+1$ with constant probability.*

Proof. We reduce this problem to a communication complexity problem. Alice holds a matrix $\boldsymbol{A} \in \mathbb{F}_p^{n\times n}$ and Bob holds a matrix $\boldsymbol{B} \in \mathbb{F}_p^{n\times n}$, where $\boldsymbol{M} = \boldsymbol{A}+\boldsymbol{B}$ and $\mathsf{rank}(\boldsymbol{M}) \in \{k, k+1\}$. Corollary 23 in [32] implies that the randomized communication complexity is $\Omega(k^2 \log p)$ to determine whether the rank of $\boldsymbol{M}$ is k or $k+1$. Alice and Bob can simulate the query algorithm using $O(\log p)$ bits of communication per query. Let $q(n,k)$ be the query complexity of this problem in the $\mathsf{u^TMv}$ model. Then $q(n,k)\log p = \Omega(k^2 \log p)$, and we conclude that $q(n,k) = \Omega(k^2)$. ◀

Now consider the real-valued version of rank testing with $\boldsymbol{M} \in \mathbb{R}^{n\times n}$. It is known that if we want to compute the rank of $\boldsymbol{M}$ up to a factor of $(1\pm\epsilon)$, then this requires $\Omega(n^{2-O(\varepsilon)})$ space in the streaming model [6]. Assadi et. al. [6] has shown that even for some special matrices of which the entries are only in $\{-1,0,1\}$, there exists an $\Omega(n^{2-O(\varepsilon)})$ space lower bound for $(1+\varepsilon)$-approximation of the rank. Notice that for $\mathsf{u^TMv}$ queries, if we choose $\boldsymbol{u} = (1,3,3^2,\ldots,3^{m-1})^{\mathrm{T}}$ and $\boldsymbol{v} = (1,3^m,3^{2m},\ldots,3^{m(m-1)})^{\mathrm{T}}$, then we can exactly reconstruct $\boldsymbol{M}$ using the value of $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$. Therefore, we assume that the matrix and the query vectors have integral values bounded by a polynomial in n. Under this assumption, we prove the following theorem:

▶ **Theorem 4.** *Given a matrix $\boldsymbol{M} \in \mathbb{R}^{n\times n}$, if we restrict that the entry of query vectors can be chosen only from $\{0,1,2,\ldots,n^c\}$ for some constant c, then $\Omega(n^{2-O(\varepsilon)})$ non-adaptive queries are necessary to obtain a $(1+\varepsilon)$-estimation of $\mathsf{rank}(\boldsymbol{M})$.*

Proof. Let $q(n)$ be the number of $\mathsf{u^TMv}$ queries sufficient to estimate the rank up to a factor of $(1\pm\epsilon)$. Consider a streaming model with updates of the form $\boldsymbol{u}_1^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}_1$, $\boldsymbol{u}_2^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}_2$, …, $\boldsymbol{u}_{q(n)}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}_{q(n)}$, where $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are the i^{th} queries made in the $\mathsf{u^TMv}$ model for $i = 1,2,\ldots,q(n)$. We can store these queries using $O(q(n)\cdot\log n)$ bits of space (as the matrix and vector entries are polynomially bounded). Using the previous results of [6], we see that $\Omega(n^{2-O(\varepsilon)})$ bits of space are necessary. This implies that $q(n)O(\log n) = \Omega(n^{2-O(\varepsilon)})$, and hence, $q(n) = \Omega(n^{2-O(\varepsilon)}/\log n) = \Omega(n^{2-O(\varepsilon)})$. ◀

3.3 Trace Estimation

Estimating the trace of a matrix presents a simple problem where $\mathsf{u^TMv}$ queries are just as powerful as matrix-vector queries, even though the latter obtains much more information per query. Sun et. al. [42] proves an $\Omega(n/\log n)$ lower bound for trace estimation of symmetric matrix, of which the entries are in $\{0,1,2,\ldots,n^3\}$, using matrix-vector queries. Since $\boldsymbol{M}\boldsymbol{v}$ contains all information of $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$, it is also a lower bound for the $\mathsf{u^TMv}$ model. Of course, n queries suffice to obtain all the diagonal elements of matrix $\boldsymbol{M}$, i.e., $\mathrm{tr}(\boldsymbol{M}) = \sum_{i=1}^n M_{ii} = \sum_{i=1}^n \boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ is the i^{th} standard basis vector. Thus, for trace estimation, we obtain an $\Omega(n/\log n)$ lower bound and an $O(n)$ upper bound. We formalize this as the following theorem.

▶ **Theorem 5.** *Let $\boldsymbol{M}$ be an $n\times n$ matrix over $\mathbb{R}$ with entries in $\{0,1,2,\ldots,n^3\}$. Assume the query vectors have entries in $\{0,1,2,\ldots,n^c\}$ for a constant $c>0$. Computing a constant factor approximation to the trace $tr(\boldsymbol{M})$ has query complexity between $\Omega(n/\log n)$ and $O(n)$.*

3.4 Deciding if a Matrix is Diagonal

In this section, we show that over any field $\mathbb{F}$, $O(\log(\frac{1}{\varepsilon}))$ queries suffice to test whether a matrix is a diagonal matrix with error probability at most $\varepsilon \in (0,1)$. Actually, it is equivalent that 1 $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ query can test with constant error probability.

For each test, we randomly and uniformly choose a subset S of $[n] = \{1, 2, 3, \ldots, n\}$ with size $|S| = \frac{n}{2}$. We select a subset G of size $|G| = 2$ from $\mathbb{F}$. Construct the query vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ as follows. For each $i \in [n]$, if $i \in S$, then let u_i be randomly and uniformly sampled from G, and $v_i = 0$; otherwise let v_i be randomly and uniformly sampled from G and $u_i = 0$. If $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v} = 0$, then output 'Success', otherwise output 'Fail'. The whole algorithm outputs 'Success' iff every test outputs 'Success'. The proof of Theorem 6 is presented in Appendix A.

▶ **Theorem 6.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over any field $\mathbb{F}$. Then with $O(\log(\frac{1}{\varepsilon}))$ queries, one can test whether $\boldsymbol{M}$ is a diagonal matrix with probability at least $1 - \varepsilon$.*

3.5 Deciding if a Matrix is Symmetric

Sun et. al. [42] has shown an $O(\log(\frac{1}{\varepsilon}))$ query upper bound in the matrix-vector model to test whether an $n \times n$ matrix $\boldsymbol{M}$ is symmetric with probability $1 - \epsilon$. We can simulate their method in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model. To do so, repeat the following process $O(\log(\frac{1}{\varepsilon}))$ times: choose two random vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ and test whether $\boldsymbol{u}^\mathrm{T}\boldsymbol{M}\boldsymbol{v} = \boldsymbol{v}^\mathrm{T}\boldsymbol{M}\boldsymbol{u}$. Using the previous results [42], we see that the error probability is at most ε. We formalize this as follows:

▶ **Theorem 7.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over any field $\mathbb{F}$. Then with $O(\log(\frac{1}{\varepsilon}))$ queries, one can test whether $\boldsymbol{M}$ is a symmetric matrix with probability at least $1 - \varepsilon$.*

3.6 Deciding if a Matrix is Unitary

The results on query complexity in this subsection also apply for testing if a matrix is orthonormal over $\mathbb{R}$, since orthonormal is a special case of unitary.

3.6.1 Randomized Queries

Given an $n \times n$ complex matrix $\boldsymbol{M}$, a single matrix-vector query can determine whether $\boldsymbol{M}$ is unitary with probability one [42]. Hence in the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model, n randomized queries suffice, by obtaining the entries of the vector $\boldsymbol{M}\boldsymbol{v}$ using $\boldsymbol{u} = \boldsymbol{e}_i$ for $1 \le i \le n$.

Now we show that the $O(n)$ algorithm is nearly optimal by proving a lower bound $\Omega(n/\log n)$ in random case.

▶ **Theorem 8.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over $\mathbb{C}$. Then to determine whether $\boldsymbol{M}$ is a unitary matrix with a constant probability, the lower bound of query complexity is $\Omega(n/\log n)$ and the upper bound is $O(n)$.*

Proof. WLOG, let $n = 2^k$, and then this problem can be reduced to Disjointness. Suppose Alice has a string $x \in \{0,1\}^n$, and Bob has a string $y \in \{0,1\}^n$. Moreover, $\boldsymbol{x}$ and $\boldsymbol{y}$ both contain exactly $\frac{n}{4}$ ones, i.e. $|\{i \in [n] \mid x_i = 1\}| = \frac{n}{4}$ and $|\{i \in [n] \mid y_i = 1\}| = \frac{n}{4}$. Now Alice and Bob want to find whether there exists an index i such that $x_i = y_i = 1$. The communication complexity of this problem is $\Omega(n)$ [7, 28]. Now we show a protocol of the communication. First, let's recall one construction of a Hadamard matrix.

▶ **Definition 9.** *Let $\boldsymbol{H}_1 = \begin{bmatrix} 1 \end{bmatrix}$, and $\boldsymbol{H}_{2^k} = \begin{bmatrix} \boldsymbol{H}_{2^{k-1}} & \boldsymbol{H}_{2^{k-1}} \\ \boldsymbol{H}_{2^{k-1}} & -\boldsymbol{H}_{2^{k-1}} \end{bmatrix}$ be a Hadamard matrix, then we define $\boldsymbol{G}_{2^k} = \frac{1}{\sqrt{2^k}}\boldsymbol{H}_{2^k}$ for any $k \ge 1$.*

By the definition, $\boldsymbol{G}_{n/4}^{*}\boldsymbol{G}_{n/4} = \boldsymbol{I}_{n/4}$, which means $\boldsymbol{G}_{n/4}$ is a unitary matrix. Also, we denote the element of row i and column j of matrix $\boldsymbol{G}_{n/4}$ by $g_{i,j}$. Then Alice constructs an $n \times n$ matrix $\boldsymbol{X}$ with the following method. Let a_i denote the i^{th} smallest position of string x of which the value is 1. For example, if $x = 00100010$, then $a_1 = 3, a_2 = 7$. Then Alice fills exactly $(n/4) \times (n/4)$ elements of matrix $\boldsymbol{X}$, i.e.

$$\boldsymbol{X}_{a_i,a_j} = \begin{cases} g_{i,j} & ,i \neq j \\ g_{i,j} - 1 & ,i = j, \end{cases} \quad \text{where } 1 \leq i,j \leq n/4.$$

Other elements of $\boldsymbol{X}$ are all 0s. Alice then constructs another matrix $\boldsymbol{X}'$, which is the same as $\boldsymbol{X}$ except that

$$\boldsymbol{X}'_{a_i,a_j} = \begin{cases} -g_{i,j} & ,i \neq j \\ -g_{i,j} - 1 & ,i = j, \end{cases} \quad \text{where } 1 \leq i,j \leq n/4.$$

Let $\boldsymbol{X}_1 = \boldsymbol{X}$ and $\boldsymbol{X}_2 = \boldsymbol{X}'$. Bob uses the similar method to construct matrices $\boldsymbol{Y}_1$ and $\boldsymbol{Y}_2$ using his string y. If x and y are not intersected, i.e. there does not exist an index k, such that $x_k = y_k = 1$, then the four matrices $\boldsymbol{M}_{ij} = \boldsymbol{X}_i + \boldsymbol{Y}_j + \boldsymbol{I}$ are all unitary for $1 \leq i,j \leq 2$, where $\boldsymbol{I}$ is the identity matrix. However, if x and y intersects, then there exists an index k such that $x_k = y_k = 1$, so there exists $i,j \in \{1,2\}$, such that the element of k^{th} row and k^{th} column of matrix $\boldsymbol{M}_{ij}$ equals

$$(-\frac{1}{\sqrt{n/4}} - 1) + (-\frac{1}{\sqrt{n/4}} - 1) + 1 = -1 - \frac{2}{\sqrt{n/4}} < -1,$$

so $\boldsymbol{M}_{ij}$ is not unitary.

Therefore, Alice and Bob can compute $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}_{ij}\boldsymbol{v}$ by sending $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}_i\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}_j\boldsymbol{v}$, which take $O(\log n)$ bits by one communication. Assume that $q(n)$ queries can determine whether an $n \times n$ matrix is unitary, since Disjointness requires $\Omega(n)$ bits, $q(n)O(\log n) = \Omega(n)$, which demonstrates that $q(n) = \Omega(n/\log n)$. We summarize the above results and obtain the following theorem. ◀

3.6.2 Deterministic Queries

For deterministic case, a trivial upper bound is $O(n^2)$ by retrieving all the entries of matrix $\boldsymbol{M}$ one by one. Now we show a strong lower bound $\Omega(n^2/\log n)$, which demonstrates that the trivial algorithm can nearly perform the best.

▶ **Theorem 10.** *Let $\boldsymbol{M}$ be an $n \times n$ matrix over $\mathbb{C}$. Then to determine whether $\boldsymbol{M}$ is a unitary matrix in deterministic case, the lower bound of query complexity is $\Omega(n^2/\log n)$.*

Proof. We reduce the problem to Disjointness. Without loss of generality, let $n = 2^k$. One can calculate easily that there are $n^- = \frac{n(n-1)}{2}$ (-1)s in matrix $\boldsymbol{H}_n$, and $n^+ = \frac{n(n+1)}{2}$ 1s. We denote the element of row i and column j of matrix $\boldsymbol{H}_n$ by $h_{i,j}$. Then, let $\boldsymbol{Z}$ be the matrix of $n \times n$, which satisfies that

$$Z_{i,j} = \begin{cases} -1, & h_{i,j} = -1 \\ 0, & h_{i,j} = 1. \end{cases}$$

Denote the positions in which elements are 1s in $\boldsymbol{H}_n$ by 1, 2, 3, ..., n^+ respectively. Now Alice holds a string x and Bob holds a string y, where $x,y \in \{0,1\}^{n^+}$. Each of them has exactly $\frac{n^+}{2}$ 1s. In the deterministic case, it requires $\Omega(n^+) = \Omega(n^2)$ bits of communication to decide whether the two strings intersect. Alice then constructs a matrix $\boldsymbol{X}$ of $n \times n$, and initially, all the entries of $\boldsymbol{X}$ are zero. Next, for each i, where $1 \leq i \leq n^+$, if $x_i = 1$, then Alice fills in 1 at position i in $\boldsymbol{X}$. Bob constructs a matrix $\boldsymbol{Y}$ with the same method. Let $\boldsymbol{M} = \frac{1}{\sqrt{n}}(\boldsymbol{X} + \boldsymbol{Y} + \boldsymbol{Z})$. Then x and y do not intersect if and only if $\boldsymbol{M}$ is unitary. To exchange $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$ needs only $O(\log n)$ bits, so the lower bound is $\Omega(n^2/\log n)$. ◀

4 Statistics Problems

4.1 All Ones Column

Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix. Sun et. al. [42] show a lower bound of $\Omega(n/\log n)$ for matrix-vector queries over $\mathbb{R}$ when restricting the entries in the query vector to $[n^c] = \{1, 2, \ldots, n^c\}$ for some constant c. This lower bound can be applied directly to the $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ model. The following theorem shows that this is tight (up to logarithmic factors) and the proof is presented in Appendix A.

▶ **Theorem 11.** *Given a matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ over $\mathbb{R}$, then $O(n)$ queries suffice to test whether there exists an all ones column in $\boldsymbol{M}$ with probability one.*

4.2 Identical Columns

Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$. Rearrange $\boldsymbol{M}$ in the following way: $\boldsymbol{M} = \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_n \end{bmatrix}$. Our task is to determine whether there exists i, j, such that $1 \le i < j \le n$ and $\boldsymbol{c}_i = \boldsymbol{c}_j$.

We consider the lower bound of query complexity over $\mathbb{F}_2$ first.

▶ **Theorem 12.** *Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix over $\mathbb{F}_2$. Let ε be a real number such that $0 < \varepsilon < 1$ and $n \ge 2(1 + \log \frac{n^2}{\varepsilon})$, then $\Omega(n)$ queries are necessary to detect whether there exist two identical columns in $\boldsymbol{M}$ with probability at least $1 - \varepsilon$.*

Proof. We reduce this problem to Disjointness. Assume Alice has a string $x \in \{0,1\}^{n-1}$, and Bob has a string $y \in \{0,1\}^{n-1}$. Now Alice could construct a matrix $\boldsymbol{X} \in \{0,1\}^{\frac{n}{2}\times n}$, where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} & \boldsymbol{a}_1 & \boldsymbol{a}_2 & \cdots & \boldsymbol{a}_{\frac{n}{2}-1} \\ 1 & 1 & 1 & \cdots & 1 \end{bmatrix}^{\mathrm{T}}.$$

We denote the $j^{\,\mathrm{th}}$ element of vector $\boldsymbol{a}_i$ as a_{ij}. For each a_{ij}, when $x_j = 1$, we let $a_{ij} = 1$; and when $x_j = 0$, we let a_{ij} be a random variable drawn from a uniform distribution in $\{0,1\}$. Bob constructs $\boldsymbol{Y}$ by the same method. Then let

$$\boldsymbol{M} = \begin{bmatrix} \boldsymbol{X} \\ \boldsymbol{Y} \end{bmatrix}$$

be an $n \times n$ matrix. If x and y intersect, then the corresponding column of $\boldsymbol{M}$ is all ones. Since the last column of $\boldsymbol{M}$ is also all ones, $\boldsymbol{M}$ contains two identical columns. If x and y do not intersect, then for every two columns, the probability that they are identical is at most $\frac{1}{2^{\frac{n}{2}-1}}$. By a union bound, the probability that there exist two identical columns is less than $\binom{n}{2} \times \frac{1}{2^{\frac{n}{2}-1}} \le \frac{n^2}{2^{\frac{n}{2}-1}} \le \varepsilon$, since $n \ge 2(1 + \log \frac{n^2}{\varepsilon})$. Alice and Bob must communicate $\Omega(n)$ bits, and sending $\boldsymbol{u}^\mathrm{T}\boldsymbol{X}\boldsymbol{v}$ and $\boldsymbol{u}^\mathrm{T}\boldsymbol{Y}\boldsymbol{v}$ each need only one bit over $\mathbb{F}_2$, so $\Omega(n)$ queries are necessary to detect two identical columns. ◀

For the upper bounds over $\mathbb{F}_2$ and $\mathbb{R}$ we have the following theorem.

▶ **Theorem 13.** *Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix.*

- *$O(n \log(n/\varepsilon))$ queries over $\mathbb{F}_2$ suffice to detect two identical columns with probability $1-\varepsilon$.*
- *$O(n)$ queries over $\mathbb{R}$ suffice to detect two identical columns with probability one.*

Proof. We choose a random n-dimensional vector $\boldsymbol{u}$, where each u_i is independent. Over $\mathbb{R}$, let u_i be chosen from a standard normal distribution $N(0,1)$; and over $\mathbb{F}_2$, let u_i be chosen uniformly from $\{0,1\}$. Notice that n queries suffice to obtain $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle$ for $1 \le i \le n$, where $\boldsymbol{c}_i$ is the i^{th} column of $\boldsymbol{M}$. If there are two identical columns $\boldsymbol{c}_i$ and $\boldsymbol{c}_j$, then $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ always holds.

Now we analyze the probability that $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ holds for two columns that are not equal. For convenience, let $\boldsymbol{v} = \boldsymbol{c}_i - \boldsymbol{c}_j$. Since $\boldsymbol{c}_i \neq \boldsymbol{c}_j$, we know that $\boldsymbol{v} \neq \boldsymbol{0}$. Assume $v_k \neq 0$ for some index k such that $1 \le k \le n$. When querying over $\mathbb{R}$, we have that

$$\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle - \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_i - \boldsymbol{c}_j \rangle = \langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{i=1}^{n} u_i v_i = u_k v_k + \sum_{i \neq k} u_i v_i.$$

Since $u_k \sim N(0,1)$ and $v_k \neq 0$, we have that $u_k v_k + \sum_{i \neq k} u_i v_i = 0$ with probability 0, which means that $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle \neq \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ with probability one. Therefore, $O(n)$ queries suffice over $\mathbb{R}$ to detect identical columns with probability one.

Working over the field $\mathbb{F}_2$, we see that $u_k v_k + \sum_{i \neq k} u_i v_i = 0$ with probability $\frac{1}{2}$. This means that $\langle \boldsymbol{u}, \boldsymbol{c}_i \rangle = \langle \boldsymbol{u}, \boldsymbol{c}_j \rangle$ with probability $\frac{1}{2}$. If we choose $\log(n^2/\varepsilon) = O(\log(n/\varepsilon))$ independent vectors $\boldsymbol{u}$, then this equality holds for every $\boldsymbol{u}$ with probability $\frac{\varepsilon}{n^2}$. Since there are $\binom{n}{2} \le n^2$ pairs (i,j), the overall error probability is less than $\frac{\varepsilon}{n^2} \cdot n^2 = \varepsilon$ by a union bound. Therefore, the query complexity in the $\mathsf{u^T Mv}$ model over $\mathbb{F}_2$ is $O(n \log(n/\varepsilon))$. ◀

4.3 Majority

Given an $n \times n$ matrix $\boldsymbol{M}$ over $\mathbb{F}_2$, we consider computing the *column-wise majority* of $\boldsymbol{M}$. That is, for each column, we compute whether it contains at least $n/2$ ones or not. We prove that $\Theta(n^2)$ queries are necessary and sufficient, even for randomized algorithms.

▶ **Theorem 14.** *Let $\boldsymbol{M} \in \mathbb{F}_2^{n \times n}$ be a binary matrix. Computing the column-wise majority of $\boldsymbol{M}$ requires $\Omega(n^2)$ queries, even for constant success probability.*

Proof. We reduce this problem to Disjointness. Assume Alice has n binary strings of length n, i.e. $x_1, x_2, \ldots, x_n$, each of which contains exactly $\frac{n}{4}$ 1s. Bob has n binary strings of length n, i.e. $y_1, y_2, \ldots, y_n$, each of which contains exactly $\frac{n}{4}$ 1s as well. We define $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ as follows:

$$f(x,y) = \begin{cases} 0, & x \text{ and } y \text{ have non-empty intersection,} \\ 1, & \text{otherwise.} \end{cases}$$

By a direct sum theorem in communication complexity, $\Omega(n^2)$ bits of communication are required to decide $(f(x_1,y_1), f(x_2,y_2), \ldots, f(x_n,y_n))$ simultaneously [34]. Let $\boldsymbol{x}_i$ be the corresponding n-dimensional column vector of string x_i. Also, let $\boldsymbol{y}_i$ be the corresponding column vector of string y_i. Alice and Bob construct matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, where

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x}_1 & \boldsymbol{x}_2 & \cdots & \boldsymbol{x}_n \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{Y} = \begin{bmatrix} \boldsymbol{y}_1 & \boldsymbol{y}_2 & \cdots & \boldsymbol{y}_n \end{bmatrix}.$$

Let $\boldsymbol{M} = \boldsymbol{X} + \boldsymbol{Y}$. Then x_i and y_i intersect if and only if the majority of i^{th} column of $\boldsymbol{M}$ is 0 since the elements are over $\mathbb{F}_2$. Furthermore, $\boldsymbol{u}^{\mathrm{T}} \boldsymbol{M} \boldsymbol{v}$ can be computed by the communication of $\boldsymbol{u}^{\mathrm{T}} \boldsymbol{X} \boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}} \boldsymbol{Y} \boldsymbol{v}$, each communication requiring one bit. Thus, the number of queries needed to decide the majority of every column is $q(n) = \Omega(n^2)$. ◀

▶ **Corollary 15.** *Given a matrix $\boldsymbol{M} \in \{0,1\}^{n \times n}$ over $\mathbb{F}_2$, then the query complexity of computing the column-wise majority is $\Theta(n^2)$, for both deterministic and randomized queries.*

4.4 Permutation Matrix

A matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ is a *permutation matrix* if each column and each row contains exactly one entry equal to 1. We consider the query complexity over both $\mathbb{R}$ and $\mathbb{F}_2$, which are very different.

We observe checking if a graph G is a perfect matching is equivalent to checking if the adjacency matrix is a permutation matrix. This also holds for the bipartite version: for a graph on $2n$ vertices, let $M_{ij} = 1$ when the i^{th} vertex on the left is connected to the j^{th} vertex on the right. The following theorem states that $O(1)$ queries suffice over the reals to check whether $\boldsymbol{M}$ is a permutation matrix with constant probability.

▶ **Theorem 16.** *Let $\boldsymbol{M} \in \{0,1\}^{n\times n}$ be a binary matrix over $\mathbb{R}$. Then, $O(1)$ queries suffice to check whether $\boldsymbol{M}$ is a permutation matrix with constant probability.*

Proof. Using a single query $\boldsymbol{u} = \boldsymbol{v} = (1,1,\ldots,1)^{\mathrm{T}}$, we first verify that $\boldsymbol{M}$ contains exactly n ones. Assume this holds. Also, assume without loss of generality that n is even.

We first describe an algorithm to test with constant probability whether each column contains a single one. Reversing the roles of columns and rows will establish the same for rows. The algorithm repeats the following process a constant number of times. Randomly select a subset $A \subseteq [n]$ of exactly $n/2$ columns. Let $\boldsymbol{u}$ be the all ones vector, and let $\boldsymbol{v} = \mathbf{1}_A$ and $\boldsymbol{v}' = \mathbf{1}_{A\setminus[n]}$ be the indicator vectors for A and its complement. Reject if either $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \neq n/2$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}' \neq n/2$.

If $\boldsymbol{M}$ is a permutation matrix, then $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}' = n/2$ holds. If $\boldsymbol{M}$ is not a permutation matrix, there must be a pair of columns (or rows), one with all zeros, and one with at least two ones. Suppose column c contains all zeros, and column c' contains at least two ones. With constant probability in choosing A, we have $c \in A$ and $c' \notin A$ or vice versa. Conditioned on this, we claim that either $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \neq n/2$ or $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}' \neq n/2$ with constant probability as well.

To see this, consider randomly partitioning the $n-2$ columns (excluding c and c') into two groups of size $\frac{n}{2}-1$. Let s_1 and s_2 be the number of ones in Groups 1 and 2, respectively. Without loss of generality, assume $s_1 \leq s_2$. Now, consider adding c and c' to the two groups, conditioned on them being separated. If c' is in Group 2, then Group 2 will have more ones than Group 1. Thus, one of the groups must not have $n/2$ ones, and our algorithm rejects with constant probability. ◀

Interestingly, the query complexity depends on the field. If $\boldsymbol{M} \in \{0,1\}^{n\times n}$ is over $\mathbb{F}_2$, then $O(1)$ queries are far from enough.

▶ **Theorem 17.** *Let $\boldsymbol{M} \in \mathbb{F}_2^{n\times n}$ be a matrix. Then, $\Omega(n)$ queries are necessary to determine whether $\boldsymbol{M}$ is a permutation matrix with constant probability.*

Proof. We reduce this problem to Disjointness. Alice holds a string $x \in \{0,1\}^n$ and Bob holds a string $y \in \{0,1\}^n$. Now Alice constructs a $3n \times 3n$ matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{A}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{A}_n \end{bmatrix} \quad \text{where} \quad \boldsymbol{A}_i = \begin{cases} \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} & \text{if } x_i = 0 \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \text{if } x_i = 1. \end{cases}$$

Bob constructs a $3n \times 3n$ matrix

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{B}_1 & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \boldsymbol{B}_2 & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \mathbf{0} & \cdots & \boldsymbol{B}_n \end{bmatrix} \quad \text{where} \quad \boldsymbol{B}_i = \begin{cases} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} & \text{if } y_i = 0 \\ \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix} & \text{if } y_i = 1. \end{cases}$$

Let $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{B}$, with addition over $\mathbb{F}_2$. Then $\boldsymbol{M}$ is a permutation matrix if and only if x and y are disjoint. Thus, the query complexity is $\Omega(n)$ since $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{B}\boldsymbol{v}$ are both a single bit. ◀

4.5 Doubly Stochastic Matrix

A non-negative real-valued matrix is *doubly stochastic* if all rows and columns sum to one.

▶ **Theorem 18.** *Let $\boldsymbol{M} \in (\mathbb{R}^+ \cup \{0\})^{n \times n}$ be a non-negative real matrix. Then $O(1)$ queries suffices to check whether $\boldsymbol{M}$ is doubly stochastic with constant probability.*

Proof. First, check whether the sum of all entries is n by choosing $\boldsymbol{u} = \boldsymbol{v} = (1, 1, \ldots, 1)^{\mathrm{T}}$ and checking whether $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = n$. Assume equality holds. If $\boldsymbol{M}$ is not doubly stochastic, then some column c (or row r) has sum > 1 and another column c' (or row r' respectively) has sum < 1. Partition the columns (rows) into two groups of size $\frac{n}{2}$. By the argument in the proof of Theorem 16, two groups sum to different values with constant probability. ◀

4.6 Matrix with Negative Entries

In our previous result for doubly stochastic matrices, we assumed that all the entries are non-negative. This assumption is necessary. If we allow negative entries in a matrix, then even checking whether or not there exists a negative entry requires $\Omega(n^2/\log n)$ queries.

▶ **Theorem 19.** *Let $\boldsymbol{M} \in \mathbb{R}^{n \times n}$ be a matrix. Then $\Omega(n^2/\log n)$ queries are necessary to test if $\boldsymbol{M}$ contains a negative entry using query vectors with entries in $\{0, \pm 1, \pm 2, \ldots, \pm n^c\}$ for a constant c.*

Proof. We reduce this problem to Disjointness. Alice holds a bit-string x with size n^2, and Bob holds a bit-string y with the same size. Alice and Bob construct $n \times n$ matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, where

$$A_{ij} = \begin{cases} 1, & x_{(i-1)n+j} = 0 \\ 0, & x_{(i-1)n+j} = 1 \end{cases} \quad \text{and} \quad B_{ij} = \begin{cases} 0, & y_{(i-1)n+j} = 0 \\ -1, & y_{(i-1)n+j} = 1 \end{cases}$$

Let $\boldsymbol{M} = \boldsymbol{A} + \boldsymbol{B}$, and notice that $\boldsymbol{M}$ contains negative entries if and only if x and y intersect. If the query complexity is $q(n)$, then by the Disjointness lower bound, $q(n) \log n = \Omega(n^2)$. ◀

5 Graph Problems

5.1 Triangle Detection

Triangle detection task means that a simple graph G is given in the form of adjacency matrix $\boldsymbol{M} \in \{0, 1\}^{n \times n}$, where n is the number of vertices in G, and we want to decide whether there exists a triangle, i.e. there exists $1 \le i < j < k \le n$, such that $M_{ij} = M_{jk} = M_{ki} = 1$. The following theorem shows a lower bound on the number of $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ queries to detect a triangle.

▶ **Theorem 20.** *Given a simple graph G consisting of n vertices in the form of its adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$, then even with a constant probability, $\Omega(n^2/\log n)$ queries are necessary to determine whether there exists a triangle in G.*

Proof. We reduce this problem to a communication complexity problem, that is [8], given a graph G with n vertices, where Alice holds some edges of G, and Bob holds the remaining edges of G, then $\Omega(n^2)$ bits of communication is required to determine whether there exists a triangle in G, even in the random case with a constant probability.

Now suppose the graph is G, and its adjacency matrix is $\boldsymbol{M} \in \{0,1\}^{n\times n}$. Alice holds some edges represented by the matrix $\boldsymbol{X}$, and Bob holds the remaining edges represented by the matrix $\boldsymbol{Y}$. Obviously $\boldsymbol{M} = \boldsymbol{X} + \boldsymbol{Y}$. Then Alice and Bob can communicate by sending $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}$ and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$, and $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ can be obtained immediately since $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{v}+\boldsymbol{u}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{v}$. Assume $q(n)$ queries can determine whether there exists a triangle, then $q(n)\log n = \Omega(n^2)$. Thus $q(n) = \Omega(n^2/\log n)$. ◀

5.2 Deciding if a Graph is a Star

Star is a special kind of tree where there exists one vertex adjacent to all the other vertices.

▶ **Theorem 21.** *$\boldsymbol{M} \in \{0,1\}^{n\times n}$ is the adjacency matrix of a simple graph G. Then $O(1)$ queries suffice to determine whether G is a star with constant probability.*

The proof of Theorem 21 is shown in Appendix A.

6 Conclusion

In this paper, we undertook an exploratory study of a new query model that considers querying a matrix through vector-matrix-vector queries. We provided new algorithms and lower bounds for problems spanning three domains: linear algebra, statistics, and graphs. For many of our results, we showed nearly matching bounds on the query complexity, sometimes up to logarithmic factors. We also demonstrated that many previously studied queries can be viewed as special cases or variants of the $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ model, and therefore, $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ queries provide a unified way to study the query complexity of various graph and matrix problems.

In terms of open questions, an interesting direction would be to identify cases where $\mathsf{u}^{\mathsf{T}}\mathsf{M}\mathsf{v}$ queries are much more efficient than previously studied models. Some options include: determining the minimum cut more efficiently than cut queries [39, 35] or estimating subgraph counts (e.g., triangles) more efficiently than local graph queries [6, 21, 40]. It could also be interesting to study the generalization of our model to k-linear forms (i.e., querying a k-tensor by specifying k vectors), comparing against k-partite independent set queries for counting k-cliques [12, 11, 19].

References

1 Hasan Abasi and Bshouty Nader. On Learning Graphs with Edge-Detecting Queries. In *Algorithmic Learning Theory*, pages 3–30, 2019.

2 Maryam Aliakbarpour, Amartya Shankha Biswas, Themis Gouleakis, John Peebles, Ronitt Rubinfeld, and Anak Yodpinyanee. Sublinear-time algorithms for counting star subgraphs via edge sampling. *Algorithmica*, 80(2):668–697, 2018. doi:10.1007/s00453-017-0287-3.

3 Noga Alon and Vera Asodi. Learning a Hidden Subgraph. *SIAM Journal on Discrete Mathematics*, 18(4):697–712, 2005.

4 Noga Alon, Richard Beigel, Simon Kasif, Steven Rudich, and Benny Sudakov. Learning a hidden matching. *SIAM Journal on Computing*, 33(2):487–501, 2004.

5 Sepehr Assadi, Michael Kapralov, and Sanjeev Khanna. A Simple Sublinear-Time Algorithm for Counting Arbitrary Subgraphs via Edge Sampling. In *Proc. 10th Innovations in Theoretical Computer Science Conference (ITCS)*, 2019.

6 Sepehr Assadi, Sanjeev Khanna, and Yang Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1723–1742. SIAM, 2017.

7 Ziv Bar-Yossef, Thathachar S Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. *Journal of Computer and System Sciences*, 68(4):702–732, 2004.

8 Ziv Bar-Yossef, Ravi Kumar, and D Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the thirteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 623–632. Society for Industrial and Applied Mathematics, 2002.

9 P. Beame, S. Har-Peled, S. Natarajan Ramamoorthy, C. Rashtchian, and M. Sinha. Edge estimation with independent set oracles. In Anna R. Karlin, editor, *Proc. Innov. Theo. Comp. Sci.* (ITCS), volume 94 of *LIPIcs*, pages 38:1–38:21, 2018.

10 I. Ben-Eliezer, T. Kaufman, M. Krivelevich, and D. Ron. Comparing the strength of query types in property testing: The case of k-colorability. *Computational Complexity*, 22(1):89–135, 2013. doi:10.1007/s00037-012-0048-2.

11 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Hyperedge estimation using polylogarithmic subset queries. *CoRR*, abs/1908.04196, 2019. arXiv:1908.04196.

12 Anup Bhattacharya, Arijit Bishnu, Arijit Ghosh, and Gopinath Mishra. Triangle estimation using tripartite independent set queries. In Pinyan Lu and Guochuan Zhang, editors, *Proc. 30th Annu. Internat. Sympos. Algorithms Comput.* (ISAAC), volume 149 of *LIPIcs*, pages 19:1–19:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ISAAC.2019.19.

13 Mark Braverman, Elad Hazan, Max Simchowitz, and Blake E. Woodworth. The gradient complexity of linear regression. *CoRR*, abs/1911.02212, 2019. arXiv:1911.02212.

14 Clément L Canonne. A survey on distribution testing: your data is big. but is it blue? In *Electronic Colloquium on Computational Complexity (ECCC)*, volume 22:63, pages 1–1, 2015.

15 Diptarka Chakraborty, Lior Kamma, and Kasper Green Larsen. Tight Cell Probe Bounds for Succinct Boolean Matrix-Vector Multiplication. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1297–1306, 2018.

16 Arkadev Chattopadhyay, Michal Koucký, Bruno Loff, and Sagnik Mukhopadhyay. Simulation Beats Richness: New Data-Structure Lower Bounds. In *Proc. 50th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 1013–1020, 2018.

17 Xi Chen, Amit Levi, and Erik Waingarten. Nearly optimal edge estimation with independent set queries. In Shuchi Chawla, editor, *Proc. 31st ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 2916–2935. SIAM, 2020. doi:10.1137/1.9781611975994.177.

18 Holger Dell and John Lapinskas. Fine-grained reductions from approximate counting to decision. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proc. 50th Annu. ACM Sympos. Theory Comput.* (STOC), pages 281–288. ACM, 2018. doi:10.1145/3188745.3188920.

19 Holger Dell, John Lapinskas, and Kitty Meeks. Approximately counting and sampling small witnesses using a colourful decision oracle. In Shuchi Chawla, editor, *Proc. 31st ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 2201–2211. SIAM, 2020. doi:10.1137/1.9781611975994.135.

20 Zeev Dvir, Alexander Golovnev, and Omri Weinstein. Static Data Structure Lower Bounds Imply Rigidity. In *Proc. 51st Annual ACM SIGACT Symp. on Theory of Computing (STOC)*, pages 967–978, 2019.

21 T. Eden, A. Levi, D. Ron, and C. Seshadhri. Approximately counting triangles in sublinear time. *SIAM J. Comput.*, 46(5):1603–1646, 2017. doi:10.1137/15M1054389.

22 T. Eden, D. Ron, and C. Seshadhri. On Approximating the Number of k-cliques in Sublinear Time. *CoRR*, abs/1707.04858, July 2017. arXiv:1707.04858.

23 Yonina C Eldar and Gitta Kutyniok. *Compressed sensing: theory and applications*. Cambridge university press, 2012.

24 U. Feige. On sums of independent random variables with unbounded variance and estimating the average degree in a graph. *SIAM J. Comput.*, 35(4):964–984, 2006. doi:10.1137/s0097539704447304.

25 O. Goldreich and D. Ron. Approximating average parameters of graphs. *Random Struct. Algo.*, 32(4):473–493, 2008. doi:10.1002/rsa.v32:4.

26 Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.

27 M. Gonen, D. Ron, and Y. Shavitt. Counting stars and other small subgraphs in sublinear-time. *SIAM J. Discrete Math*, 25(3):1365–1411, 2011. doi:10.1137/100783066.

28 Johan Håstad and Avi Wigderson. The randomized communication complexity of set disjointness. *Theory of Computing*, 3(1):211–219, 2007.

29 Bala Kalyanasundaram and Georg Schintger. The probabilistic communication complexity of set intersection. *SIAM Journal on Discrete Mathematics*, 5(4):545–557, 1992.

30 Kasper Green Larsen and Ryan Williams. Faster Online Matrix-Vector Multiplication. In *Proc. 28th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2182–2189, 2017.

31 Yi Li, Huy L Nguyen, and David P Woodruff. On approximating matrix norms in data streams. *SIAM Journal on Computing*, 48(6):1643–1697, 2019.

32 Yi Li, Xiaoming Sun, Chengu Wang, and David P Woodruff. On the communication complexity of linear algebraic problems in the message passing model. In *International Symposium on Distributed Computing*, pages 499–513. Springer, 2014.

33 Yi Li and David P. Woodruff. Tight bounds for sketching the operator norm, schatten norms, and subspace embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016, Paris, France*, pages 39:1–39:11, 2016.

34 Marco Molinaro, David P Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *Proceedings of the twenty-fourth annual ACM-SIAM symposium on Discrete algorithms*, pages 1738–1756. SIAM, 2013.

35 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted Min-Cut: Sequential, Cut-Query and Streaming Algorithms. In *Proc. 52nd Annual ACM Symposium on Theory of Computing (STOC)*, 2020.

36 Sivaramakrishnan Natarajan Ramamoorthy and Cyrus Rashtchian. Equivalence of Systematic Linear Data Structures and Matrix Rigidity. In *Proc. 11th Innovations in Theoretical Computer Science Conference (ITCS)*, 2020.

37 K. Onak, D. Ron, M. Rosen, and R. Rubinfeld. A near-optimal sublinear-time algorithm for approximating the minimum vertex cover size. In *Proc. 23rd ACM-SIAM Sympos. Discrete Algs.* (SODA), pages 1123–1131, 2012. URL: http://dl.acm.org/citation.cfm?id=2095116.2095204.

38 AA Razborov. On the distributional complexity of disjointness. *Theoretical Computer Science*, 106:385–390, 1992.

39 Aviad Rubinstein, Tselil Schramm, and S Matthew Weinberg. Computing exact minimum cuts without knowing the graph. In *Proc. 9th Innovations in Theoretical Computer Science Conference (ITCS)*, 2018.

40 C. Seshadhri. A simpler sublinear algorithm for approximating the triangle count. *CoRR*, abs/1505.01927, May 2015. arXiv:1505.01927.

41 Max Simchowitz, Ahmed El Alaoui, and Benjamin Recht. Tight query complexity lower bounds for PCA via finite sample deformed wigner law. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 1249–1259, 2018.

42 Xiaoming Sun, David P Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. In *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*, 2019.

43 Dan Wang and Zhu Han. *Sublinear algorithms for big data applications*. Springer, 2015.

44 J. Wang, E. Lo, and M. L. Yiu. Identifying the most connected vertices in hidden bipartite graphs using group testing. *IEEE Tran. Knowl. Data Eng.*, 25(10):2245–2256, 2013. doi:10.1109/TKDE.2012.178.

45 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

46 Andrew Chi-Chin Yao. Probabilistic computations: Toward a unified measure of complexity. In *18th Annual Symposium on Foundations of Computer Science*, pages 222–227. IEEE, 1977.

A Remaining Proofs

Proof of Theorem 6. We show that for each query, if $\boldsymbol{M}$ is a diagonal matrix, then the test will always succeed; if not, then the test will fail with constant probability. Then by error reduction, $O(\log(\frac{1}{\varepsilon}))$ queries suffice to achieve error probability at most ε.

For each query, we choose $\boldsymbol{u}$, $\boldsymbol{v}$ as the above algorithm describes. Therefore,

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \sum_{i\in S, j\in[n]-S} u_i v_j M_{ij}.$$

If $\boldsymbol{M}$ is diagonal, then $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = 0$ always holds. If $\boldsymbol{M}$ is not diagonal, then we claim that $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ is non-zero with constant probability. In this case, there exists an off-diagonal element $M_{k\ell} \neq 0$ with $k \neq \ell$. With probability at least $\frac{1}{4}$, $k \in S$ and $\ell \in [n] \setminus S$ simultaneously. Conditioning on this event, let $t_i = \sum_{j\in[n]-S} v_j M_{ij}$, and rewrite $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v}$ as

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \sum_{i\in S} u_i \sum_{j\in[n]-S} v_j M_{ij} = \sum_{i\in S} u_i t_i.$$

Since

$$t_p = \sum_{j\in[n]\setminus S} v_j M_{kj} = v_\ell M_{k\ell} + \sum_{j\in[n]\setminus S\setminus\{\ell\}} v_j M_{kj} \triangleq v_\ell M_{k\ell} + T,$$

and $M_{k\ell} \neq 0$, $v_\ell M_{k\ell}$ has two different possible values. Moreover, at most one choice satisfies $v_\ell M_{k\ell} + T = 0$. So $t_k \neq 0$ with probability at least $\frac{1}{2}$. Now under the condition that $t_k \neq 0$,

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} = \sum_{i\in S} u_i t_i = u_k t_k + \sum_{i\in S\setminus\{k\}} u_i t_i \triangleq u_k t_k + R.$$

Since $t_k \neq 0$, by the same argument, $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{v} \neq 0$ with probability at least $\frac{1}{2}$. Combining all of these events, the test fails with probability at least $\frac{1}{16}$, which completes the proof. ◀

Proof of Theorem 11. We construct a random vector $\boldsymbol{u} \in \mathbb{R}^n$, where each entry u_i is independent and follows the standard Gaussian distribution. Let $\boldsymbol{e}_i$ denote the n dimensional vector with i^{th} entry 1 and all other entries 0s, and let $\boldsymbol{e} = \sum_{i=1}^n \boldsymbol{e}_i$ be the all ones n-dimensional vector. Also, let $\boldsymbol{c}_i$ denote the i^{th} column of matrix $\boldsymbol{M}$. Since we have

$$\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i = \boldsymbol{u}^{\mathrm{T}} \begin{bmatrix} \boldsymbol{c}_1 & \boldsymbol{c}_2 & \cdots & \boldsymbol{c}_n \end{bmatrix} \boldsymbol{e}_i = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{c}_i = \sum_{j=1}^n u_j c_{ij},$$

if we compute the sum of all entries of $\boldsymbol{u}$, i.e., $s = \sum_{i=1}^n u_i$, then when $\boldsymbol{c}_i$ is an all ones column, all $c_{ij} = 1$, so $s = \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$. Otherwise,

$$s - \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i = \sum_{1\le j\le n, c_{ij}=0} u_j.$$

The above quantity equals to 0 with probability 0, which means $s \neq \boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$ with probability one. By querying $\boldsymbol{u}^{\mathrm{T}}\boldsymbol{M}\boldsymbol{e}_i$ for $1 \le i \le n$, and comparing the result to s, we can detect whether there is an all ones column with probability one using $O(n)$ queries. ◀

Proof of Theorem 21. First, check whether $\boldsymbol{M}$ contains exact $2(n-1)$ ones. If not, $\boldsymbol{M}$ is obviously not a star. Now we assume that $\boldsymbol{M}$ contains exact $2(n-1)$ ones, which means G contains $(n-1)$ edges. Then equally divide the vertices into 2 groups of size $\frac{n}{2}$ randomly and uniformly. We only need to check whether the sum of degrees in one group is $\frac{n}{2}$, and another $\frac{3n}{2}-2$. If this is true, the algorithm should report that $\boldsymbol{G}$ is a star. Otherwise, the algorithm reports that $\boldsymbol{G}$ is not a star. We prove this algorithm has a constant error probability.

If G is a star, then the sum of degrees in one group is $\frac{n}{2}$, and another $\frac{3n}{2}-2$.

If G is not a star, we claim that there exists two vertices v_1 and v_2 with different degrees, which satisfies $|\deg(v_1) - \deg(v_2)| < n-2$.

Since G is not a star, then the degree of any vertex can be at most $n-2$.

- If there exists a vertex v_1 with degree $n-2$, then when n is large (e.g. $n > 10$), there must exists another vertex v_2 with degree 1. Therefore,

  $$|\deg(v_1) - \deg(v_2)| = (n-2) - 1 = n-3 < n-2.$$

- If there does not exists a vertex with degree $n-2$, then the degree of all vertices are in $\{0, 1, 2, \ldots, n-3\}$. When n is large enough (e.g. $n > 10$), there must exist two vertices v_1 and v_2 with different degrees, and

  $$|\deg(v_1) - \deg(v_2)| \le (n-3) - 0 = n-3 < n-2.$$

Now with probability at least $\frac{1}{2}$, v_1 and v_2 are in the different groups. Without loss of generality, assume $\deg(v_1) > \deg(v_2)$. Conditioned on this, we can decompose the random partition procedure into 2 steps. First, we randomly and uniformly partition the other $n-2$ vertices except v_1 and v_2 into 2 groups with size $\frac{n}{2}-1$. Assume in Group 1 the sum of degrees in these vertices is s_1 and in Group 2 the sum is s_2. Without loss of generality, assume $s_1 \le s_2$. The second step is to place v_1 in Group 1, v_2 in Group 2, or v_2 in Group 2, v_1 in Group 1 both with probability $\frac{1}{2}$.

If we have that

$$\begin{aligned} s_1 + \deg(v_1) &= \tfrac{3n}{2} - 2, & s_2 + \deg(v_2) &= \tfrac{n}{2}, \\ s_1 + \deg(v_2) &= \tfrac{n}{2}, & s_2 + \deg(v_1) &= \tfrac{3n}{2} - 2 \end{aligned}$$

hold simultaneously, then $\deg(v_1) - \deg(v_2) = n-2$, a contradiction.

If

$$\begin{aligned} s_1 + \deg(v_1) &= \tfrac{n}{2}, & s_2 + \deg(v_2) &= \tfrac{3n}{2} - 2, \\ s_1 + \deg(v_2) &= \tfrac{n}{2}, & s_2 + \deg(v_1) &= \tfrac{3n}{2} - 2, \end{aligned}$$

then $\deg(v_1) = \deg(v_2)$, a contradiction.

Also, since $s_1 \le s_2, \deg(v_1) > \deg(v_2)$ and $\frac{3n}{2} - 2 \ge \frac{n}{2}$, it is impossible that $s_1 + \deg(v_2) = \frac{3n}{2} - 2$ and $s_2 + \deg(v_1) = \frac{n}{2}$ both holds.

So with probability at least $\frac{1}{2}$, the sums of the two groups will not be $\frac{3n}{2} - 2$ and $\frac{n}{2}$.

Overall, with probability at least $\frac{1}{4}$, the sums of the two groups will not be $\frac{3n}{2} - 2$ and $\frac{n}{2}$. ◀

A.1 Local Graph Queries and Estimating Subgraph Counts

The following four local graph queries can be implemented by $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries.

▶ **Lemma 22.** *Given the adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ of a simple graph $G = (V, E)$, then the following four queries:*

- ***Degree query*** *i: the degree of vertex i.*
- ***Neighbor query*** *(i, j): the j^{th} neighbor of vertex i.*
- ***Pair query*** *(i, j): whether the edge (i, j) exists.*
- ***Edge-sample query****: sample an edge e uniformly at random from E.*

can be implemented by $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries.

As one application, we mention the problem of counting subgraphs. Given the adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ of a simple graph G, we want to estimate the number of occurrences of H in G, where H is a given subgraph (such as a triangle). Assadi et. al. [5] shows that with

$$\widetilde{O}\left(\min\left\{m, \frac{m^{\rho(H)}}{\#H}\right\}\right)$$

of the above four standard graph queries, we can obtain a $(1\pm\varepsilon)$-approximation to the number of occurrences of H in G with high probability. Here, $\#H$ is the number of occurrences of H in G, m is the number of edges, and $\rho(H)$ is the fractional edge-cover of H. Also, the $\widetilde{O}(\cdot)$ notation ignores ε and $\log n$ terms, as well as the size of graph H. By Lemma 22, the four standard graph queries can be implemented by $O(\log n)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries. Therefore, we derive the following result.

▶ **Proposition 23.** *Given the adjacency matrix $\boldsymbol{M} \in \{0,1\}^{n\times n}$ of a simple graph G and an arbitrary small target graph H, $\widetilde{O}\left(min\left\{m, \frac{m^{\rho(H)}}{\#H}\right\}\right)$ $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries suffice to obtain a $(1 \pm \varepsilon)$-approximation to the number of occurrences of H in G with high probability.*

We briefly compare this to work on independent set queries [12, 11, 19]. Proposition 23 achieves a general result for $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries, whereas estimating triangles or other subgraphs with bipartite independent set queries is an open question. Moreover, estimating larger subgraphs seems to require higher-order queries (e.g., tripartite independent set queries). This suggests that, as expected, $\mathsf{u}^\mathsf{T}\mathsf{M}\mathsf{v}$ queries may be more powerful for a variety of problems.

Almost Optimal Distribution-Free Sample-Based Testing of k-Modality



Dana Ron
Tel Aviv University, Israel
danaron@tau.ac.il

Asaf Rosin
Tel Aviv University, Israel
asaf.rosin@gmail.com

Abstract

For an integer $k \geq 0$, a sequence $\sigma = \sigma_1, \ldots, \sigma_n$ over a fully ordered set is k-modal, if there exist indices $1 = a_0 < a_1 < \cdots < a_{k+1} = n$ such that for each i, the subsequence $\sigma_{a_i}, \ldots, \sigma_{a_{i+1}}$ is either monotonically non-decreasing or monotonically non-increasing. The property of k-modality is a natural extension of monotonicity, which has been studied extensively in the area of property testing. We study one-sided error property testing of k-modality in the distribution-free sample-based model. We prove an upper bound of[1] $O\left(\frac{\sqrt{kn}\log k}{\epsilon}\right)$ on the sample complexity, and an almost matching lower bound of $\Omega\left(\frac{\sqrt{kn}}{\epsilon}\right)$. When the underlying distribution is uniform, we obtain a completely tight bound of $\Theta\left(\sqrt{\frac{kn}{\epsilon}}\right)$, which generalizes what is known for sample-based testing of monotonicity under the uniform distribution.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases Sample-based property testing, Distribution-free property testing, k-modality

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.27

Category RANDOM

Funding *Dana Ron*: Supported by the Israel Science Foundation (grant number 1146/18) and the Kadar-family award.
Asaf Rosin: This work is part of the author's MSc research.

Acknowledgements We would like to thank Ronitt Rubinfeld, since this work started as part of a project in her course, and Eden Kuperwasser, for being a great project partner. We also thank an anonymous reviewer for helpful comments.

1 Introduction

Monotonicity of functions has been studied extensively in the area of property testing [30, 35, 29, 46, 33, 32, 6, 45, 1, 37, 38, 31, 12, 16, 19, 20, 15, 24, 23, 41, 7, 21, 36, 18, 4, 26, 27, 13, 40, 42, 22, 25, 14, 43]. The different works vary in the domains and ranges they consider, as well as in the precise task studied (e.g., standard testing vs. tolerant testing and distance approximation). However, what is common to almost all of these results, is that they allow query access to the tested function, and the underlying distribution is uniform.

[1] Since our results hold for any $k \geq 0$, we should actually replace the term $\sqrt{kn}$ by $\sqrt{(k+1)n}$ and $\log k$ by $\log(k+2)$, but for the sake of readability, we refrain from doing so.


© Dana Ron and Asaf Rosin;
licensed under Creative Commons License CC-BY

In this work, we consider a natural extension of monotonicity: *k-modality*. Since the domain we study is $[n] = \{1, \ldots, n\}$, it is convenient to think of the tested object as being a sequence $\sigma = \sigma_1, \ldots, \sigma_n$, whose elements belong to a fully ordered set. A sequence σ is said to be *k-modal* if there exist indices $1 = a_0 < a_1 < \cdots < a_{k+1} = n$, such that for each i, the subsequence $\sigma_{a_i}, \ldots, \sigma_{a_{i+1}}$ is either monotonically non-decreasing or monotonically non-increasing. In other words, a sequence is k-modal if there are at most k "peaks" and "valleys" (excluding the endpoints). For example, the sequence $3, 2, 2, 1, 2, 3, 4, 2$ is 2-modal, but not 1-modal (see Figure 1). We shall assume for simplicity that σ is over $\mathbb{R}$, and use $\mathcal{M}_k$ to denote the set of all k-modal sequences over $\mathbb{R}$. Observe that a sequence is unate (i.e., either monotonically non-decreasing or monotonically non-increasing) if and only if it is 0-modal.

We study distribution-free sample-based testing of k-modality with one-sided error. Namely, the testing algorithm is given as input $k \geq 0$ and $\epsilon > 0$. For an arbitrary unknown distribution $p : [n] \to [0, 1]$, it is provided with a sample of pairs (i, σ_i), where i is selected i.i.d according to p. If σ is k-modal, then the algorithm should accept with probability 1, while if σ is ϵ-far from k-modality with respect to p, then the algorithm should reject with probability at least $2/3$. A sequence σ is ϵ-far from k-modality with respect to p, if $\sum_{i:\tau_i \neq \sigma_i} p(i) > \epsilon$ for every k-modal sequence $\tau = \tau_1, \ldots, \tau_n$.

Thus, the algorithm cannot select the symbols of σ that it observes (as in the case when queries are allowed), and it must work for every underlying distribution p. Since we require that the testing algorithm have one-sided error, it may reject σ only if the sample contains evidence that σ is not k-modal. Therefore, the question we address is:

> What is the minimum sample size $s = s(n, k, \epsilon)$ such that for every sequence σ (of length n) and underlying distribution p, if σ is ϵ-far from being k-modal with respect to p, then the sample contains evidence that σ is not k-modal?

It is known that for monotonicity and unateness (i.e., the special case of $k = 0$), when the underlying distribution p is uniform, then a sample of size $\Theta(\sqrt{n/\epsilon})$ is both necessary and sufficient.[2] The question is how does the sample complexity increase as k increases, and what is the effect of having a general underlying distribution p.[3]

1.1 Our results

Our first result is the following upper bound for distribution-free testing.[4]

▶ **Theorem 1.1.** *The sample complexity of distribution-free one-sided error sample-based testing of k-modality is $O\left(\frac{\sqrt{kn}\log k}{\epsilon}\right)$.*

We show that the upper bound in Theorem 1.1 is almost tight (up to a factor of $\log k$), by establishing the next lower bound.

▶ **Theorem 1.2.** *The sample complexity of distribution-free one-sided error sample-based testing of k-modality is $\Omega\left(\frac{\sqrt{kn}}{\epsilon}\right)$. This lower bound holds for any $\epsilon < 1/4$ and $k \leq n/4 - 1$.*

---

[2] The lower bound for monotonicity can be found for example in [36, Claim 7.5.1] (where it actually holds for two-sided error), and this extends to unateness. The upper bound is folklore, where a more general statement, regarding any poset, follows from [33, Theorems 6&14].

[3] The problem of distribution-free sample-based testing of monotonicity is also considered in [36, Claim 7.5.2]. The upper bound they claim is correct in terms of the dependence on n (which is $\sqrt{n}$), but not in terms of the dependence on $1/\epsilon$. In our analysis, we fix the flaw in their argument.

[4] See footnote 1.

When the underlying distribution is uniform, we obtain a completely tight result, which generalizes the known result for testing monotonicity. Here, rather than having a linear dependence on $1/\epsilon$, the complexity grows like $1/\sqrt{\epsilon}$.

▶ **Theorem 1.3.** *The sample complexity of one-sided error sample-based testing of k-modality under the uniform distribution is $\Theta\left(\sqrt{\frac{kn}{\epsilon}}\right)$. The lower bounds holds for any $\epsilon < 1/4$ and $k \leq \epsilon n$.*

Note that the requirement that $k \leq \epsilon n$ is not really a constraint as we are only interested in the sublinear regime.

1.2 Techniques

The lower bounds. We start by shortly discussing our lower bounds (Theorem 1.2 for the distribution-free case, and the lower bound in Theorem 1.3, for the uniform case). They are both variants of the known lower bound for testing monotonicity with one-sided error under the uniform distribution. In both lower bounds, the sequence σ is of the form $2, 1, 4, 3, \ldots, 2m, 2m-1, 3m, \ldots, 3m$ (where the value $3m$ appears $n-2m$ times), for an appropriate choice of m. When the underlying distribution is uniform, m is set to be $2\epsilon n$ (assuming, for simplicity, that $2\epsilon n$ is an integer). In the distribution-free case, m is set to be $n/2-1$, and the underlying distribution p assigns weight $\frac{2\epsilon}{m} = \frac{4\epsilon}{n-2}$ to each of the first $2m = n-2$ symbols, and weight $\frac{1-4\epsilon}{2}$ to each of the remaining two symbols. In both cases it is not hard to verify that σ is ϵ-far from being k-modal (with respect to the corresponding distribution). What is also common to both cases is that in order to obtain evidence that σ is not k-modal, it is necessary that the sample "hit" at least $k/2$ pairs of indices $(2i-1, 2i)$ for $i \in [m]$. By a birthday-paradox-type argument, the sample must be of size $\Omega\left(\sqrt{km}/\epsilon\right)$. The lower bounds now diverge due to the difference in the setting of m, where the flexibility of the distribution-free case allows us to set m to be $\Theta(n)$ and obtain a higher lower bound.

The upper bound for the distribution-free case. We first observe that in order to test k-modality, it suffices to test two closely related properties. For the sake of simplicity, here we describe and discuss one of them, which we denote by $\mathcal{F}_t^{\Uparrow}$ where $t = k+3$. A sequence $\sigma = \sigma_1, \ldots, \sigma_n$ belongs to $\mathcal{F}_t^{\Uparrow}$ if and only if there is no subsequence of indices $x_1, \ldots, x_t$ such that $1 \leq x_1 < \cdots < x_t \leq n$ and such that $\sigma_{x_i} < \sigma_{x_{i+1}}$ for every odd i and $\sigma_{x_i} > \sigma_{x_{i+1}}$ for every even i. We show that for any distribution p, if σ is ϵ-far with respect to p from $\mathcal{F}_t^{\Uparrow}$, then a sample of size $O(\sqrt{tn}\log t/\epsilon)$ will contain such a subsequence of indices.

A central ingredient in the proof of this sample-complexity upper bound is a structural claim. It states that if a sequence σ is ϵ-far from $\mathcal{F}_t^{\Uparrow}$, then there exist t indices $1 = a_1 < a_2 < \cdots < a_t = n$ for which the following holds. For each subsequence $\sigma_{a_i}, \ldots, \sigma_{a_{i+1}}$, if i is odd, then the subsequence is at least (ϵ/t)-far (with respect to p) from being monotonically non-decreasing, and if i is even, then it is at least (ϵ/t)-far from being monotonically non-increasing. Observe that the i^{th} subsequence and the $(i+1)^{\text{th}}$ subsequence share a common symbol. This is of importance when the common symbol has relatively large weight according to p.

The next ingredient is a claim regarding the probability of obtaining evidence, for each such subsequence, concerning its non-monotonicity (in the appropriate direction). In a certain sense we are reducing the problem of testing k-modality to testing monotonicity, where there are several subtleties to address. First, when considering the task of testing monotonicity for each of these subsequences, the fact that the underlying distribution is arbitrary, means

that we need to deal both with very large probabilities and with very small probabilities, which makes the analysis more complex than in the uniform case. Second, recall that each subsequence is (ϵ/t)-far from being monotone. The "stand-alone" problem of distribution-free testing of monotonicity has sample complexity that grows linearly with the inverse of the distance to monotonicity. This seems to suggest that we get a linear dependence on t (recall that $t = k + 3$), while we claim that the dependence is $\tilde{O}(\sqrt{k})$. Therefore, the reduction to the $t - 1$ instances of testing monotonicity, should be done with care. Finally, assume that we obtain evidence of non-monotonicity (in the appropriate direction) for each subsequence (where there may be overlap between neighboring subsequences due to the common symbol). We observe that we can combine these "small pieces of evidence" to infer that σ does not belong to $\mathcal{F}_t^{\Uparrow}$.

The upper bound for the uniform case. The improvement in the sample complexity for the uniform case (as compared to the distribution-free case) has two sources. The first is that the basic task of testing monotonicity requires a smaller sample when the underlying distribution is uniform. The second is that we do not apply the structural claim described above to "break" σ into predetermined subsequences and then consider the task of testing monotonicity for each of them. Instead, the subsequences are essentially determined as part of the probabilistic analysis, together with the evidence against their monotonicity. Thus, rather than taking a union bound over events of violating monotonicity in predetermined subsequences, we lower bound the probability of sampling a sufficient number of violations by analysing an appropriate sum of geometric random variables.

Specifically, we apply the Poissonization technique (see, e.g., [48, Chapter 10]), which allows us to analyze the sample as if each pair (i, σ_i) is selected independently. We then define a process that can be viewed as traversing the sequence σ while selecting the sample "on the fly", and gathering evidence against monotonicity of subsequences. We lower bound the probability (over the selection of the sample) that the process gathers sufficient pieces of evidence before σ is fully traversed.

1.3 Related results

As noted at the start of this section, there is a plethora of works on testing monotonicity and unateness. Here we focus only on those results in which the domain is the same as ours, namely $[n]$. Unless stated explicitly otherwise, the results are for testing with queries and under the uniform distribution.

Monotonicity testing over $[n]$ was first studied by Ergun et al. [30]. They gave an algorithm whose query complexity is $O(\log n/\epsilon)$. They also showed that $\Omega(\log n)$ queries are necessary for any non-adaptive comparison-based algorithm and constant ϵ. Fischer [32] proved that this lower bound actually holds for adaptive algorithms as well.

For the special case of binary sequences, k-modality is essentially the same as k-monotonicity, which was studied in [17]. To be precise, k-monotonicity of binary sequences is equivalent to $\mathcal{F}_{k+2}^{\Uparrow}$ (where $\mathcal{F}_{k+2}^{\Uparrow}$ is as defined in Section 1.2). They show that it is possible to test k-monotonicity by performing $O(k/\epsilon)$ (non-adaptive) queries. Furthermore, they prove that any one-sided error (possibly adaptive) tester for k-monotonicity over $[n]$ must have query complexity $\Omega(k/\epsilon)$. If two-sided error is allowed, then no dependence on k is necessary, and the query complexity is $\mathrm{poly}(1/\epsilon)$ [17, 14].

The related property of k-interval functions over $[0, 1]$ was studied in [39, 3]. Each such function is defined by a partition of $[0, 1]$ into (at most) k intervals, where on each interval the value of the function is constant (either 0 or 1). Observe that if we consider a discretized

version of this property where the domain is $[n]$, then it is the same as $(k-2)$-modality. Balcan et al. [3] (strengthening the result of [39]) give an upper bound of $\sqrt{k} \cdot \text{poly}(1/\epsilon)$ on the sample complexity of testing this property under the uniform distribution with two-sided error. We note that if queries are allowed, then the dependence on k can be removed [39, 3]. In addition, the result can be generalized to the distribution-free case in the active testing model [3].[5]

Other related works on testing (with queries) that generalize testing monotonicity include [8] for testing local properties of d-dimensional arrays and [44, 9, 10] for testing forbidden order patterns.

Distribution-free testing of monotonicity (with queries) was studied in [37], approximating the distance of a sequence to monotonicity was studied in [45, 2], and the problem of testing monotonicity of distributions (over totally ordered domains) was addressed in [5].

1.4 Organization

We start with some general preliminaries in Section 2. In Section 3 we present several definitions and observations relating to k-modality, which are later applied in our analysis. The upper bound for distribution-free testing is provided in Section 4, and the one for the uniform distribution in Section 5. Both lower bounds are given in Appendix C.

2 Preliminaries: sequences, distances and property testing

For an integer n, let $[n] = \{1, \ldots, n\}$, and for two integers $i \leq j$, let $[i, j] = \{i, \ldots, j\}$. For a sequence $\sigma = \sigma_1 \ldots \sigma_n$ and a subset of indices $Q \subseteq [n]$, we use $\sigma_{|Q}$ to denote the subsequence of σ corresponding to the indices in Q. A *property* of sequences $\mathcal{P}$ is simply a set of sequences. We say that σ *has property* $\mathcal{P}$, or that σ *satisfies* $\mathcal{P}$, if $\sigma \in \mathcal{P}$.

In what follows, we present several notions and notations that are defined based on a probability distribution $p : [n] \to [0, 1]$. For a set $S \subseteq [n]$, let $p(S) = \sum_{i \in S} p(S)$. Whenever p is clear from the context, we refer to $p(S)$ as the *probability weight* of S, or simply the *weight* of S.

The *distance* between σ and τ *with respect to* p, denoted $\text{dist}(\sigma, \tau, p)$, is $\sum_{i:\sigma_i \neq \tau_i} p(i)$ (or ∞ if they are not of the same length). For a property $\mathcal{P}$, the distance of σ from $\mathcal{P}$ with respect to p, denoted $\text{dist}(\sigma, \mathcal{P}, p)$, is $\min_{\tau \in \mathcal{P}}\{\text{dist}(\sigma, \tau, p)\}$. We say that σ is ϵ-far from $\mathcal{P}$ with respect to p, or (more concisely) that (σ, p) is ϵ-far from $\mathcal{P}$, if $\text{dist}(\sigma, \mathcal{P}, p) > \epsilon$. Otherwise it is ϵ-close.

▶ **Definition 2.1.** *A distribution-free sample-based testing algorithm for a property $\mathcal{P}$ of sequences is given parameters n and ϵ as well as access to samples from an unknown sequence σ of length n, generated according to an unknown distribution $p : [n] \to [0, 1]$. Namely, it receives pairs (i, σ_i) where i is distributed i.i.d. according to p. The algorithm should satisfy the following:*

[5] The notion of *active testing* is an adaptation of the notion of *active learning* to the context of property testing. The algorithm is given an unlabeled sample distributed according to the underlying distribution p and it may query the labels of part of the sample points. The two complexity measures of interest are hence the (unlabeled) sample complexity, and the query complexity (number of queries performed on the sample points). When the latter equals the former, this coincides with sample-based (distribution-free) testing (referred to as *passive testing* in [3]). However, one may aim at performing fewer queries at the cost of a larger number of unlabeled samples.

- *If $\sigma \in \mathcal{P}$, then the algorithm should accept with probability at least $2/3$.*
- *If $\mathrm{dist}(\sigma, \mathcal{P}, p) > \epsilon$, then the algorithm should reject with probability at least $2/3$.*

If p is known to be the uniform distribution over $[n]$, then we simply say that the algorithm is a sample-based testing algorithm.

If the algorithm always accepts sequences that have property $\mathcal{P}$, then we say that it has one-sided error. otherwise it has two-sided error. The sample-complexity of the algorithm is the number of samples it views (as a function of ϵ and n) when performing the aforementioned task.

In the context of testing k-modality (which is a special case of testing properties that are determined by some parameter), the algorithm is also provided with the parameter k, and its sample complexity may depend on k.

▶ **Definition 2.2.** *For positive integers n and s, and a distribution $p : [n] \to [0, 1]$, we let $I_n(s, p)$ denote the random variable corresponding to a set [6] consisting of s indices from $[n]$ that are selected i.i.d. according to p.*

Note that in order to obtain an upper bound s on the sample complexity of distribution-free sample-based one-sided error testing for a hereditary property $\mathcal{P}$ of sequences,[7] it suffices to show that $\Pr[\sigma_{|I_n(s,p)} \notin \mathcal{P}] \geq 2/3$, for every distribution $p : [n] \to [0, 1]$ and for every sequence σ of length n that is ϵ-far from $\mathcal{P}$ with respect to p.

It will also be useful to define an additional distance measure, which we refer to as the *deletion distance*. Let σ be a sequence of length n and $p : [n] \to [0, 1]$ a weight function (so that $\sum_{i=1}^{n} p(i)$ is not necessarily 1). For a property of sequences $\mathcal{P}$ and a subset $R \subseteq [n]$, let $\mathrm{del}(\sigma, \mathcal{P}, p, R)$ denote the minimum, taken over subsets $D \subseteq R$ such that $\sigma_{|R \setminus D} \in \mathcal{P}$, of $p(D)$ (if there is no such D, then $\mathrm{del}(\sigma, \mathcal{P}, p, R) = \infty$). If $R = [n]$, then we use the shorthand $\mathrm{del}(\sigma, \mathcal{P}, p)$ for $\mathrm{del}(\sigma, \mathcal{P}, p, [n])$.

3 Definitions and observations for k-modality

In this subsection we introduce several notions that will be used in our analysis.

For a pair of indices $x, y \in [n]$ such that $x < y$, we say that (x, y) is an *ascent* (*descent*) with respect to a sequence $\sigma = \sigma_1, \ldots, \sigma_n$, if $\sigma_x < \sigma_y$ ($\sigma_x > \sigma_y$). A pair (x, y) is an *up-pair*, denoted $\Uparrow$-pair (*down-pair*, denoted $\Downarrow$-pair) if it is an ascent (descent). We say that (x, y) and (x', y') are disjoint if $\{x, y\} \cap \{x', y'\} = \emptyset$. For $\Updownarrow \in \{\Uparrow, \Downarrow\}$, we denote $\mathrm{inv}(\Uparrow) = \Downarrow$ and $\mathrm{inv}(\Downarrow) = \Uparrow$.

The next notion will aid us in characterizing k-model sequences by being "free" of certain patterns.

▶ **Definition 3.1.** *Let σ be a sequence of length n, let t be an integer, and let $1 \leq x_1 < \cdots < x_t \leq n$. We say that $(x_1, \ldots, x_t)$ is a t-upward (t-downward) subsequence with respect to σ if for every odd $i \in [t-1]$, (x_i, x_{i+1}) is an ascent (descent), and for every even $i \in [t-1]$, (x_i, x_{i+1}) is a descent (ascent).*
We shall use the symbolic shorthand t-$\Uparrow$ for t-upward sequences and t-$\Downarrow$ for t-downward sequences.

[6] Note that while the sampled indices i (as defined in Definition 2.1) are i.i.d. and hence may appear with repetitions, $I_n(s)$ is a set and hence does not include repetitions

[7] A property of sequences is hereditary if it is preserved under restrictions to subsequences.

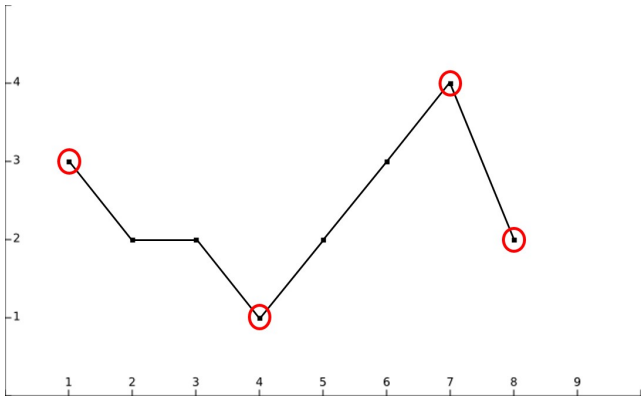


Figure 1 An illustration of the sequence $\sigma = 3,2,2,1,2,3,4,2$. Notice that the marked subsequence $(1,4,7,8)$ is a 4-downward (4-$\Downarrow$) subsequence with respect to σ, and that $\sigma \in \mathcal{F}_5^{\Downarrow}$.

▶ **Definition 3.2.** *For $\Updownarrow \in \{\Uparrow, \Downarrow\}$, we denote by $\mathcal{F}_t^{\Updownarrow}$ the set of all sequences σ such that there is no t-$\Updownarrow$ subsequence with respect to σ (so that $\mathcal{F}$ stands for "free").*
For the special case of $t = 2$, we say that a sequence is $\Updownarrow$-monotone if it belongs to $\mathcal{F}_2^{\mathrm{inv}(\Updownarrow)}$.

For example, if $\sigma = 3,2,2,1,2,3,4,2$, then $(1,4,7,8)$ is a 4-downward (4-$\Downarrow$) subsequence with respect to σ, and $\sigma \in \mathcal{F}_5^{\Downarrow}$. For an Illustration, see Figure 1.

Observe that a sequence σ is k-modal if and only if there is no $(k+3)$-$\Uparrow$ subsequence nor any $(k+3)$-$\Downarrow$ subsequence with respect to σ:

▶ **Observation 3.3.** *For any non-negative integer k we have that $\mathcal{M}_k = \mathcal{F}_{k+3}^{\Uparrow} \bigcap \mathcal{F}_{k+3}^{\Downarrow}$.*

The next notion and observation will be useful when analyzing the evidence found in a sample, that a sequence is not k-modal.

▶ **Definition 3.4.** *Let σ be a sequence of length n, let t be a positive integer, and let $1 \leq x_1 < y_1 \leq x_2 < y_2 \leq \cdots \leq x_t < y_t$. We say that $((x_1,y_1),\ldots,(x_t,y_t))$ is a t-upward-pair (t-downward-pair) sequence with respect to σ, if for every odd $i \in [t]$, (x_i,y_i) is an ascent (descent), and for every even $i \in [t]$, (x_i,y_i) is a descent (ascent). Here too we use the symbolic shorthands t-$\Uparrow$-pair (for t-upward-pair sequences) and t-$\Downarrow$-pair (for t-downward-pair sequences).*

▶ **Observation 3.5.** *For any sequence σ, integer $t \geq 2$, and $\Updownarrow \in \{\Uparrow, \Downarrow\}$, if there is a $(t-1)$-$\Updownarrow$-pair sequence with respect to σ, then there is a t-$\Updownarrow$ subsequence with respect to σ.*

To verify the validity of the last observation, consider for simplicity the case that $\Updownarrow = \Uparrow$. Given a $(t-1)$-$\Uparrow$-pair sequence $((x_1,y_1),\ldots,(x_{t-1},y_{t-1}))$ we define a t-$\Uparrow$ subsequence $(x'_1,\ldots,x'_t)$ as follows: $x'_1 = x_1$, $x'_t = y_{t-1}$, $x'_i = \max(y_{i-1},x_i)$ for each even $i \in [2,t-1]$, and $x'_i = \min(y_{i-1},x_i)$ for each odd $i \in [2,t-1]$.

We shall also make use of the following observation, which will allow us to work with the deletion distance. Its simple proof is given in Appendix B.

▶ **Observation 3.6.** *Let σ be a sequence of length n, $p : [n] \to [0,1]$ a probability distribution, and k a non-negative integer. Then $\mathrm{dist}(\sigma, \mathcal{M}_k, p) = \mathrm{del}(\sigma, \mathcal{M}_k, p)$.*

The last observation in this section will allow us to reduce the problem of testing $\mathcal{M}_k$ to testing $\mathcal{F}_{k+3}^{\Uparrow}$ and $\mathcal{F}_{k+3}^{\Downarrow}$.

▶ **Observation 3.7.** *Let σ be a sequence of length n, $p : [n] \to [0,1]$ a probability distribution, and k a non-negative integer. Then* $\mathrm{dist}(\sigma, \mathcal{M}_k, p) \leq \mathrm{del}(\sigma, \mathcal{F}^{\Uparrow}_{k+3}, p) + \mathrm{del}(\sigma, \mathcal{F}^{\Downarrow}_{k+3}, p)$.

To verify Observation 3.7, let $t = k+3$ and for each $\Updownarrow \in \{\Uparrow, \Downarrow\}$ let $D^{\Updownarrow} \subseteq [n]$ be a subset satisfying $\sigma_{|[n]\setminus D^{\Updownarrow}} \in \mathcal{F}^{\Updownarrow}_t$ and $p(D^{\Updownarrow}) = \mathrm{del}(\sigma, \mathcal{F}^{\Updownarrow}_t, p)$. Since $\mathcal{F}^{\Uparrow}_t$ and $\mathcal{F}^{\Downarrow}_t$ are hereditary properties, for $D = D^{\Uparrow} \cup D^{\Downarrow}$, we have that $\sigma_{|[n]\setminus D} \in \mathcal{F}^{\Uparrow}_t \cap \mathcal{F}^{\Downarrow}_t$. By Observation 3.3, this means that $\sigma_{|[n]\setminus D} \in \mathcal{M}_k$, so that $\mathrm{del}(\sigma, \mathcal{M}_k, p) \leq p(D^{\Updownarrow}) \leq p(D^{\Uparrow}) + p(D^{\Downarrow}) = \mathrm{del}(\sigma, \mathcal{F}^{\Uparrow}_t, p) + \mathrm{del}(\sigma, \mathcal{F}^{\Downarrow}_t, p)$. By Observation 3.6, $\mathrm{dist}(\sigma, \mathcal{M}_k, p) = \mathrm{del}(\sigma, \mathcal{M}_k, p)$ and Observation 3.7 is verified.

4 The upper bound for distribution-free testing

In this section we prove Theorem 1.1, which is restated next.

▶ **Theorem 1.1.** *The sample complexity of distribution-free one-sided error sample-based testing of k-modality is $O\left(\frac{\sqrt{kn}\log k}{\epsilon}\right)$.*

4.1 Structural claims

We start with two structural claims, where the second builds on the first (and where the first will also serve us for the upper bound under the uniform distribution).

▷ Claim 4.1. Let $\sigma = \sigma_1, \ldots, \sigma_n$ be a sequence of length n and $p : [n] \to [0,1]$ a weight function. Then for any $m \in [n]$, $t > 2$ and $\Updownarrow \in \{\Uparrow, \Downarrow\}$,

$$\mathrm{del}(\sigma, \mathcal{F}^{\Updownarrow}_t, p) \leq \mathrm{del}(\sigma, \mathcal{F}^{\Updownarrow}_2, p, [m]) + \mathrm{del}(\sigma, \mathcal{F}^{\mathrm{inv}(\Updownarrow)}_{t-1}, p, [m+1, n]) \ .$$

In order to prove this claim, we essentially show that if it is possible to partition a sequence into two parts, such that the first is free of upward pairs, and the second is free of $(t-1)$-downward sequences, then the entire sequence is free of t-upward sequences.

Proof. By the definition of the deletion distance, there exists a set $I_1 \subseteq [m]$ of weight $\mathrm{del}(\sigma, \mathcal{F}^{\Updownarrow}_2, p, [m])$, such that $\sigma_{|[m]\setminus I_1} \in \mathcal{F}^{\Updownarrow}_2$. Similarly, there exists a set $I_2 \subseteq [m+1, n]$ of weight $\mathrm{del}(\sigma, \mathcal{F}^{\mathrm{inv}(\Updownarrow)}_{t-1}, p, [m+1, n])$, such that $\sigma_{|[m+1,n]\setminus I_2} \in \mathcal{F}^{\mathrm{inv}(\Updownarrow)}_{t-1}$.

Let $\tilde{\sigma}$ denote the sequence that is obtained from σ by deleting all the indices in $I_1 \cup I_2$. Namely, $\tilde{\sigma} = \sigma_{[n]\setminus I_1 \cup I_2}$. We next show that $\tilde{\sigma} \in \mathcal{F}^{\Updownarrow}_t$. We represent $\tilde{\sigma}$ as a concatenation of two sequences, τ_1 and τ_2, where $\tau_1 = \sigma_{|[m]\setminus I_1}$ and $\tau_2 = \sigma_{|[m+1,n]\setminus I_2}$. Assume, contrary to the claim, that there exists a t-$\Updownarrow$ subsequence $(x_1, x_2 \ldots, x_t)$ with respect to $\tilde{\sigma}$. Since (x_1, x_2) is a $\Updownarrow$-pair (by the definition of a t-$\Updownarrow$ subsequence) while $\tau_1 \in \mathcal{F}^{\Updownarrow}_2$, necessarily $x_2 > |\tau_1|$. But then $(x_2, \ldots, x_t)$ is a $(t-1)$-inv($\Updownarrow$) subsequence with respect to τ_2 (more precisely, $(x_2 - |\tau_1|, \ldots, x_t - |\tau_1|)$ is such a subsequence), in contradiction to $\tau_2 \in \mathcal{F}^{\Updownarrow}_{t-1}$.

We conclude that

$$\mathrm{del}(\sigma, \mathcal{F}^{\Updownarrow}_t, p) \leq p(I_1 \cup I_2) = \mathrm{del}(\sigma, \mathcal{F}^{\Updownarrow}_2, p, [m]) + \mathrm{del}(\sigma, \mathcal{F}^{\mathrm{inv}(\Updownarrow)}_{t-1}, p, [m+1, n]) \ ,$$

as claimed. ◁

We next recursively apply Claim 4.1 to show that if σ is far from $\mathcal{F}^{\Updownarrow}_t$, then we can define $t-1$ (almost disjoint) consecutive subsequences, such that each is relatively far from either $\mathcal{F}^{\Updownarrow}_2$ or $\mathcal{F}^{\mathrm{inv}(\Updownarrow)}_2$.

▷ Claim 4.2. Let σ be a sequence of length n, $p : [n] \to [0,1]$ a weight function, $t \geq 2$, and $\Updownarrow \in \{\Uparrow, \Downarrow\}$. Denote $\Delta = \mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p)$ and suppose that $\Delta > 0$. Then there exist indices $1 = a_1 < \cdots < a_t = n$ such that $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, [a_i, a_{i+1}]) \geq \Delta/(t-1)$ for every odd $i \in [t-1]$, and $\mathrm{del}(\sigma, \mathcal{F}_2^{\mathrm{inv}(\Updownarrow)}, p, [a_i, a_{i+1}]) \geq \Delta/(t-1)$ for every even $i \in [t-1]$.

Note that partitioning the domain into disjoint intervals that obey the condition stated in Claim 4.2 may not be possible, due to the existence of indices with a large weight according to p. To address this issue we allow each pair of consecutive intervals ($[a_i, a_{i+1}]$ and $[a_i, a_{i+1}]$) to share a common index. This suffices for our purposes as we shall see in Section 4.3

Proof. We prove the claim by induction on t. For the base case, $t = 2$, the claim is trivial. Turning to the induction step, we assume that the claim holds for $t-1$ (where $t > 2$), and prove it for t.

Define a_2 to be the smallest index in $[n]$ satisfying the required condition $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, [a_2]) \geq \Delta/(t-1)$ (recall that Δ denotes $\mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p)$). Note that: (1) such an index exists, as $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, [n]) \geq \mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p, [n]) = \Delta \geq \Delta/(t-1)$, and (2) $a_2 > 1$, as $\sigma_1 \in \mathcal{F}_2^{\Updownarrow}$ so that $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, \{1\}) = 0$.

Thus, we can apply Claim 4.1 with $m = a_2 - 1$ to obtain that $\mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p) \leq \mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, [a_2 - 1]) + \mathrm{del}(\sigma, \mathcal{F}_{t-1}^{\mathrm{inv}(\Updownarrow)}, p, [a_2, n])$. By the definition of a_2, we have that $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, [a_2 - 1]) < \Delta/(t-1)$. We infer that $\mathrm{del}(\sigma, \mathcal{F}_{t-1}^{\mathrm{inv}(\Updownarrow)}, p, [a_2, n]) > \Delta - \Delta/(t-1) = (t-2)\Delta/(t-1)$.

The claim is established because the existence of $a_3, \ldots, a_t$ (recall that $t > 2$) satisfying the required conditions is implied by the induction hypothesis, using $\tilde{t} = t - 1$, $\tilde{\Updownarrow} = \mathrm{inv}(\Updownarrow)$, $\widetilde{\sigma} = \sigma_{a_2} \ldots \sigma_n$ and $\widetilde{\Delta} = (t-2)\Delta/(t-1)$. Notice that the conditions are satisfied, as $\widetilde{\Delta}/(t-2) = \Delta/(t-1)$. ◁

4.2 Probabilistic claims – obtaining evidence of non-monotonicity

In the following claim we give conditions under which a sample contains evidence that a subsequence is not in $\mathcal{F}_2^{\Updownarrow}$ (with probability at least 2/3). We later apply this claim to the subsequences defined in Claim 4.2.

▷ Claim 4.3. Let σ be a sequence of length n, $p : [n] \to [0,1]$ a probability distribution, and R a subset of $[n]$. Suppose that for $\Updownarrow \in \{\Uparrow, \Downarrow\}$ and for positive β and δ, we have that $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, R) \geq \beta$, and that $p(i) \geq \delta$ for each $i \in R$. Then for $s = \Theta(1/\sqrt{\delta \cdot \beta})$, the probability over the choice of $Q = I_n(s, p)$ that $\sigma_{|R \cap Q} \notin \mathcal{F}_2^{\Updownarrow}$, is at least 2/3.

In order to prove Claim 4.3, we lower bound the probability weight of a sample that falls into a prespecified subset of the domain $[n]$.

▷ Claim 4.4. Let $p : [n] \to [0,1]$ be a probability distribution, and C a subset of $[n]$. Suppose that for positive β and $\delta \leq \beta/c$, where c is a sufficiently large constant, $p(C) \geq \beta$ and $p(x) \geq \delta$ for each $x \in C$. Then for $s = 1/\sqrt{\delta \cdot \beta}$, letting $Q = I_s(n, p)$,

$$\Pr_Q \left[p(C \cap Q) \geq \frac{\delta \beta s}{4} \right] \geq \frac{9}{10} .$$

Note that if Q and $C \cap Q$ were defined as multisets rather than sets, i.e., if we were to take repetitions into account, then Claim 4.4 would have followed from a standard tail bound (Fact A.3). However, since in our case $C \cap Q$ is a set, we need to analyse the effect of collisions in the sample. As the distribution p may contain large probabilities, the collisions can have a significant impact. In order to overcome this difficulty, we use a "flattening technique" that is similar to the one introduced in [28] (see also [34]).

Proof. We prove the claim in two stages. In the first stage we define a "flattened" probability distribution $\hat{p}$ over $[\hat{n}]$ for $\hat{n} \geq n$ together with a subset $\widehat{C} \subseteq [\hat{n}]$, for which we show the following: for any positive integer s, letting $\widehat{Q} = I_s(\hat{n}, \hat{p})$, we have that

$$\text{for any } \tau \geq 0, \quad \Pr_Q[p(C \cap Q) \geq \tau] \geq \Pr_{\widehat{Q}}[\hat{p}(\widehat{C} \cap \widehat{Q}) \geq \tau] \,. \tag{1}$$

In the second stage we show, using certain properties of $\hat{p}$ and $\widehat{C}$, that for s as in the premise of the claim,

$$\Pr_{\widehat{Q}}\left[\hat{p}(\widehat{C} \cap \widehat{Q}) \geq \frac{\delta\beta s}{4}\right] \geq \frac{9}{10} \,. \tag{2}$$

Stage I. If $p(x) \in [\delta, 2\delta]$ for each $x \in C$, then we simply let $\hat{n} = n$, $\hat{p} = p$, and $\widehat{C} = C$, so that Equation (1) holds trivially. Otherwise, the high level idea is that we "split" each $x \in C$ into a subset of indices, each with probability weight in $[\delta, 2\delta]$ (according to $\hat{p}$), as explained precisely next. Assume, without loss of generality, that $C = \{m+1, \ldots, n\}$ for some $m < n-1$. For each $x \in C$ let $\alpha(x) = \lceil \frac{p(x)}{2\delta} \rceil$. Set $\hat{n} = m + \sum_{x=m+1}^{n} \alpha(x)$ and $\widehat{C} = \{m+1, \ldots, \hat{n}\}$. It remains to define $\hat{p}$. With each $x \in C$ we associate a (disjoint) subset, denoted $J(x)$, of $\alpha(x)$ indices in $\{m+1, \ldots, \hat{n}\}$. In particular, we can let $J(x) = \{m+1+\sum_{x'=m+1}^{x-1} \alpha(x'), \ldots, m+\sum_{x'=m+1}^{x} \alpha(x')\}$ for each $x \in \{m+1, \ldots, n\}$. It will be useful to define a mapping $\phi : [\hat{n}] \to [n]$, where for each $y \in \widehat{C}$, $\phi(y) = x$ where x satisfies $y \in J(x)$. For $y \notin \widehat{C}$, we let $\phi(y) = y$. For each $y \notin C$ we set $\hat{p}(y) = p(y)$, and for each $y \in \widehat{C} = \{m+1, \ldots, \hat{n}\}$ we set $\hat{p}(y) = p(\phi(y))/\alpha(\phi(y))$. Note that by the definition of $\alpha(\cdot)$, this ensures that $\hat{p}(y) \in [\delta, 2\delta]$ and that $p(x) = \hat{p}(J(x))$ for each $x \in C$.

In order to establish Equation (1), we apply a coupling argument. Specifically, we define a random variable $\widetilde{Q} \subseteq [n]$ based on $\widehat{Q} \subseteq [\hat{n}]$ as follows: $\widetilde{Q} = \{\phi(y) : y \in \widehat{Q}\}$. Since for each $x \in [n]$ we have that $p(x) = \sum_{y:\phi(y)=x} \hat{p}(y)$, by its definition, $\widetilde{Q}$ is identically distributed to Q. Next observe that for each $x \in C$, if $x \notin \widetilde{Q}$, then necessarily $J(x) \cap \widehat{Q} = \emptyset$, while if $x \in \widetilde{Q}$, then $p(x) \geq \hat{p}(J(x) \cap \widehat{Q})$. Therefore, $p(C \cap \widetilde{Q}) \geq \hat{p}(\widehat{C} \cap \widehat{Q})$, and Equation (1) follows.

Stage II. Since $\hat{p}(\widehat{C}) = p(C)$, and $p(C) \geq \beta$, we have that $\hat{p}(\widehat{C}) \geq \beta$. For $s = 1/\sqrt{\delta\beta}$, let $Y_1, \ldots, Y_s$ be independent random variables such that for each $y \in [\hat{n}]$, $\Pr[Y_r = y] = \hat{p}(y)$. Let $\hat{s} = |\{r : Y_r \in \widehat{C} \setminus \{Y_1, \ldots, Y_{r-1}\}\}|$, and observe that $\hat{p}(\widehat{C} \cap \widehat{Q}) \geq \hat{s} \cdot \delta$ (since $\hat{p}(y) \geq \delta$ for each $y \in \widehat{C}$). Hence, in order to establish Equation (2), it suffices to upper bound the probability that $\hat{s} < \beta s/4$.

To this end we also define $s' = |\{r : Y_r \in \widehat{C}\}|$, so $\mathbb{E}[s'] = \hat{p}(\widehat{C}) \cdot s$. We may think of $\hat{s}$ as being determined by first determining s', and then taking s' samples from $\widehat{C}$, where each $y \in \widehat{C}$ is selected independently with probability $\hat{p}(y)/\hat{p}(\widehat{C})$. Since $s' \sim \mathrm{Bin}(s, \hat{p}(\widehat{C}))$, we have (by Fact A.3) that $\Pr[s' < \mathbb{E}[s']/2] \leq e^{-\hat{p}(\widehat{C})s/8}$. By our setting of s and since $\delta \leq \beta/c$, this probability is at most $1/20$ for a sufficiently large constant c. We henceforth condition on the event that $s' \geq \mathbb{E}[s']/2 \geq \beta s/2$. If all Y_r that belong to C were distinct, then we would have that $\hat{s} = s'$, and we would be done. Since this is not necessarily the case, it remains to show that $\hat{s}$ is not much smaller than s'. To be precise, since our goal is to lower bound the probability that $\hat{s} < \beta s/4$, we condition on the event that $s' = \beta s/2$ (as the probability that $\hat{s} < \beta s/4$ can only decrease as s' increases).

Let $q = |\{r : Y_r \in \widehat{C} \cap \{Y_1, \ldots, Y_{r-1}\}\}|$, so that $\hat{s} = s' - q$. Observe that $\mathbb{E}[q] \leq \binom{s'}{2} \cdot \frac{2\delta}{\hat{p}(\widehat{C})} \leq s' \cdot \frac{\delta s'}{\hat{p}(\widehat{C})}$, which by our condition on s' and since $\hat{p}(\widehat{C}) \geq \beta$ is at most $\frac{\beta s}{2} \cdot \frac{\delta s}{2}$. Once again by our setting of s and since $\delta \leq \beta/c$, this is at most $\frac{\beta s}{80}$ for a sufficiently large constant c. By Markov's inequality, $\Pr\left[q > \frac{\beta s}{4}\right] \leq 1/20$. We can conclude that with probability at least $1 - 2/20 = 9/10$, $\hat{s} \geq \beta s/4$, and the claim follows. ◁

We are now ready to prove Claim 4.3, which applies Claim 4.4 to a vertex cover C in the "violation graph", as defined next.

Proof of Claim 4.3. For indices $x, y \in R$, we say that (x, y) is a "violating pair" if $x < y$ and $\sigma_x \sigma_y \notin \mathcal{F}_2^{\Updownarrow}$. Consider the "violation graph" $G_R = ([n], E_R)$, where E_R is the set of all violating pairs, and recall that a *vertex cover* of a graph is a subset C of vertices such that each edge of the graph contains at least one vertex in C. For a subset of vertices S, let $\Gamma_R(S)$ denote the set of neighbors of vertices belonging to S in the graph G_R.

Let C be a minimum-weight vertex cover of G_R with respect to p. Note that $\sigma_{|R\setminus C} \in \mathcal{F}_2^{\Updownarrow}$, so that $p(C) = \mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, R) \geq \beta$. Let $s = 16/\sqrt{\delta \cdot \beta}$. First, consider a sample, denoted Q_1, of $s/16$ vertices (indices in $[n]$) drawn independently according to p. By Claim 4.4, the probability that $p(C \cap Q_1) \geq \beta\delta s/64$ is at least $9/10$. Conditioned on this event occurring, we consider a second sample, denoted Q_2, of $15s/16$ vertices drawn independently according to p. As observed in [36], $p(\Gamma_R(C \cap Q_1)) \geq p(C \cap Q_1)$ (since otherwise, $C' = (C \setminus Q_1) \cup \Gamma_R(C \cap Q_1)$ is a vertex cover with smaller weight than C). Hence, each of these $15s/16$ sampled vertices belongs to $\Gamma_R(C \cap Q_1)$ with probability at least $p(C \cap Q_1)$. Therefore, the probability that $Q_1 \times Q_2$ contains no violating pair is upper bounded by

$$(1 - p(C \cap Q_1))^{15s/16} \leq \left(1 - \frac{\beta\delta s}{64}\right)^{15s/16} \leq e^{-15\beta\delta s^2/1024} \leq e^{-3.75} \leq 1/10 \,.$$

We conclude that the sample Q contains a violating pair, and hence $\sigma_{|R\cap Q} \notin \mathcal{F}_2^{\Updownarrow}$, with probability at least $1 - 1/10 - 1/10 > 2/3$, as claimed. ◁

4.3 Wrapping things up

By combining Claims 4.2 and Claim 4.3, we can establish the next lemma, which gives an upper bound on the sample complexity of one-sided error distribution-free testing of $\mathcal{F}_t^{\Updownarrow}$.

▶ **Lemma 4.5.** *Let σ be a sequence of length n, $p : [n] \to [0, 1]$ a probability distribution, $t \geq 2$, $\epsilon > 0$ and $\Updownarrow \in \{\Uparrow, \Downarrow\}$. If $\mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p) > \epsilon$, then for $s = \Theta(\sqrt{tn} \log t/\epsilon)$ and $Q = I_n(s, p)$ we have that $\Pr\left[\sigma_{|R\cap Q} \notin \mathcal{F}_t^{\Updownarrow}\right] \geq \frac{2}{3}$.*

Proof. Let $\delta = \frac{\epsilon}{2n}$ and let $B = \{x \in [n] : p(x) \geq \delta\}$. Since $p([n] \setminus B) < n \cdot \delta = \epsilon/2$, we have that $\mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p, B) > \epsilon/2$. We would like to apply Claim 4.2 to $\sigma_{|B}$, and hence we need to first define a corresponding weight function (over $[|B|]$), which we denote by $p_{|B}$. Specifically, denoting the elements in B by $\{b_1, \ldots, b_{|B|}\}$ where $b_1 < \cdots < b_{|B|}$, we let $p_{|B}(j) = p(b_j)$. Observe that $\mathrm{del}(\sigma_{|B}, \mathcal{F}_t^{\Updownarrow}, p_{|B}) = \mathrm{del}(\sigma, \mathcal{F}_t^{\Updownarrow}, p, B)$.

We can now apply Claim 4.2 to $\sigma_{|B}$ and $p_{|B}$ (as well as t and $\Updownarrow$), and obtain $1 \leq a_1 < \cdots < a_t \leq |B|$ as stated in the claim. For each $i \in [t-1]$, let $R_i = B \cap [b_{a_i}, b_{a_{i+1}}]$. Therefore, $\mathrm{del}(\sigma, \mathcal{F}_2^{\Updownarrow}, p, R_i) > \epsilon/(2(t-1))$ for each odd $i \in [t-1]$, and $\mathrm{del}(\sigma, \mathcal{F}_2^{\mathrm{inv}(\Updownarrow)}, p, R_i) > \epsilon/(2(t-1))$ for each even $i \in [t-1]$.

Consider any fixed choice of $i \in [t-1]$. In particular, assume first that i is odd. Suppose we apply Claim 4.3 with $R = R_i$, $\beta = \epsilon/(2(t-1))$, $\delta = \epsilon/(2n)$ (and $\Updownarrow$). Observe that $1/\sqrt{\delta\beta} = \Theta(\sqrt{tn}/\epsilon)$. Since in the current lemma $s = \Theta(\sqrt{tn} \log t/\epsilon)$, we get that for $Q = I_n(s, p)$ (i.e., a sample of size $c \log t$ times larger than the sample in the statement of Claim 4.3), $\Pr_Q[\sigma_{|R_i\cap Q} \in \mathcal{F}_2^{\Updownarrow}] \leq (1/3)^{c \log t} < 1/(3t)$ (for an appropriate constant c). That is, $\sigma_{|R_i\cap Q}$ does not contain an $\mathrm{inv}(\Updownarrow)$-pair with probability at most $1/(3t)$. An analogous statement holds for each even $i \in [t-1]$ (with respect to $\mathcal{F}_2^{\mathrm{inv}(\Updownarrow)}$).

By applying a union bound over all $i \in [t-1]$, we get that with probability at least $2/3$, there are $(t-1)$ pairs $(x_1, y_1), \ldots, (x_{t-1}, y_{t-1})$ such that $x_i, y_i \in R_i \cap Q$, for which the following holds: For every odd $i \in [t-1]$, the pair (x_i, y_i) is a $\Updownarrow$-pair, and for every even $i \in [t-1]$, the pair (x_i, y_i) is an inv($\Updownarrow$)-pair. Notice that for each $i \in [t-1]$, $y_i \in R_i$ and $x_{i+1} \in R_{i+1}$, so we have that $y_i \leq x_{i+1}$. Therefore (recalling Definition 3.5), with probability at least $2/3$, the sample Q contains a $(t-1)$-$\Updownarrow$-pair sequence with respect to σ. By Observation 3.5, this implies that the sample contains a t-$\Updownarrow$ subsequence with respect to σ, that is, $\sigma_{|Q} \notin \mathcal{F}_t^{\Updownarrow}$. ◀

We are now ready to prove Theorem 1.1.

Proof of Theorem 1.1. If $\mathrm{dist}(\sigma, \mathcal{M}_k, p) > \epsilon$, then, by Observation 3.7, for some $\Updownarrow \in \{\Uparrow, \Downarrow\}$, $\mathrm{del}(\sigma, \mathcal{F}_{k+3}^{\Updownarrow}, p) > \epsilon/2$ (otherwise, $\mathrm{del}(\sigma, \mathcal{F}_{k+3}^{\Uparrow}, p) + \mathrm{del}(\sigma, \mathcal{F}_{k+3}^{\Downarrow}, p) \leq \epsilon$, so that by Observation 3.7, $\mathrm{dist}(\sigma, \mathcal{M}_k, p) \leq \epsilon$, and we obtain a contradiction). The theorem follows by applying Lemma 4.5 and Observation 3.3. ◀

5 The upper bound for testing under the uniform distribution

In this section we prove the upper bound of Theorem 1.3, which is restated next.

▶ **Theorem 1.3** *The sample complexity of one-sided error sample-based testing of k-modality under the uniform distribution is $\Theta\left(\sqrt{\frac{kn}{\epsilon}}\right)$. The lower bounds holds for any $\epsilon < 1/4$ and $k \leq \epsilon n$.*

Referring to the notations introduced in Section 2, when p is the uniform distribution over $[n]$, we shall use the shorthand $\mathrm{dist}(\sigma, \mathcal{P})$ for $\mathrm{dist}(\sigma, \mathcal{P}, p)$, $I_n(s)$ for $I_n(s, p)$, and $\mathrm{del}(\sigma, \mathcal{P})$ for $\mathrm{del}(\sigma, \mathcal{P}, p)$. Actually, rather than working with the weighted/normalized deletion distance $\mathrm{del}(\sigma, \mathcal{P})$, it will be convenient to work with the absolute deletion distance $\mathrm{Del}(\sigma, \mathcal{P}) = \mathrm{del}(\sigma, \mathcal{P}) \cdot n$. Namely, $\mathrm{Del}(\sigma, \mathcal{P})$ is the minimum *size* of a subset $D \subseteq [n]$ such that $\sigma_{|[n]\setminus D} \in \mathcal{P}$ (where for hereditary properties, such a subset always exists).

For the sake of the analysis, it will be useful to analyze the sample complexity of testing k-modality (with one-sided error and under the uniform distribution) when the sample is selected according to the Poisson distribution. Recall that the Poisson distribution $\mathrm{Poi}(\lambda)$ takes value $x \in \mathbb{N}$ with probability $e^{-\lambda}\lambda^x/x!$. The next definition is analogous to Definition 2.2.

▶ **Definition 5.1.** *For positive integers n and s, we use $I_n^{\mathrm{Poi}}(s)$ to denote the random variable consisting of a subset of $[n]$ such that for each $i \in [n]$ we have $\Pr[i \in I_n^{\mathrm{Poi}}(s)] = \Pr[\mathrm{Poi}(s/n) \neq 0] = 1 - e^{-s/n}$.*

The following lemma is directly implied by [11, Lemma 2.2] (which in turn refers to [47]).

▶ **Lemma 5.2.** *For any property $\mathcal{P}$, positive integers n and s, and sequence σ of length n,*

$$\Pr[\sigma_{|I_n(s)} \notin \mathcal{P}] \geq \Pr[\sigma_{|I_n^{\mathrm{Poi}}(s/2)} \notin \mathcal{P}] - \frac{4}{s}\,.$$

We next introduce a definition and a simple claim.

▶ **Definition 5.3.** *An ascent/descent (x, y) is said to* start *at x and* end *at y.*
We define the first *ascent (descent) in a sequence σ to be the ascent (descent) that ends first. In case that there are multiple such ascents (descents), choose the one that starts first.*
For an integer $r > 1$, we recursively define the r^{th} ascent (descent) in σ to be the first ascent (descent) in the subsequence obtained from σ by deleting the first $r-1$ ascents (descents).

Recall that we use the notations $\Uparrow$-pair for an ascent and $\Downarrow$-pair for a descent.

▷ Claim 5.4. Let σ be a sequence of length n, $t \geq 2$ an integer, and $\Updownarrow \in \{\Uparrow, \Downarrow\}$. Then σ contains at least $\frac{1}{2}\mathrm{Del}(\sigma, \mathcal{F}_t^{\Updownarrow})$ disjoint $\Updownarrow$-pairs.

Proof. We prove that at least $\frac{1}{2}\mathrm{Del}(\sigma, \mathcal{F}_t^{\Updownarrow})$ disjoint $\Updownarrow$-pairs exist. Delete the first $\Updownarrow$-pair (as defined in Definition 5.3), if it exists. Continue to delete the second $\Updownarrow$-pair, third $\Updownarrow$-pair, etc. as long as possible. Assume that after deleting the y^{th} $\Updownarrow$-pair, no $\Updownarrow$-pair remains. Therefore, by deleting $2y$ symbols from σ we turned it into a sequence in $\mathcal{F}_2^{\Uparrow}$, and hence also in $\mathcal{F}_t^{\Uparrow}$ (for any $t \geq 2$). Thus $y \geq \frac{1}{2}\mathrm{Del}(\sigma, \mathcal{F}_t^{\Uparrow})$, so at least $\frac{1}{2}\mathrm{Del}(\sigma, \mathcal{F}_t^{\Uparrow})$ disjoint $\Updownarrow$-pairs exist. ◁

The next structural claim is a corollary of Claim 4.1.

▷ Claim 5.5. Let σ be a sequence of length n, $t \geq 2$ an integer, $\Updownarrow \in \{\Uparrow, \Downarrow\}$, and y an integer smaller than $\frac{1}{2}\mathrm{Del}(\sigma, \mathcal{F}_t^{\Updownarrow})$. Let $m \in [n]$ be the index such that the y^{th} $\Updownarrow$-pair in σ ends at m. Then $\mathrm{Del}(\sigma, \mathcal{F}_t^{\Updownarrow}) \leq 2y + \mathrm{Del}(\sigma_{m+1}, \ldots, \sigma_n, \mathcal{F}_{t-1}^{\mathrm{inv}(\Updownarrow)})$.

We are now ready to state and prove the lemma that is the heart of our upper bound argument. As opposed to the upper-bound argument for the distribution-free case (Lemma 4.5), In the proof of Lemma 5.6 we do not apply the structural claim described above to "break" σ into predetermined subsequences and then consider the task of testing monotonicity for each of them. Instead, the subsequences are determined by a process that traverses the sequence while selecting the sample "on the fly", and gathering evidence against monotonicity of subsequences.

▶ **Lemma 5.6.** *Let $\epsilon > 0$, $t \geq 2$ an integer, and $\Updownarrow \in \{\Uparrow, \Downarrow\}$. If $\mathrm{Del}(\sigma, \mathcal{F}_t^{\Updownarrow}) > \epsilon n$, then for $s = \Theta(\sqrt{\frac{tn}{\epsilon}})$ and $Q = I_n^{\mathrm{Poi}}(s)$ we have that*

$$\Pr\left[\sigma_{|Q} \notin \mathcal{F}_t^{\Updownarrow}\right] \geq \frac{5}{6}\,.$$

Proof. Denote $\Delta = \mathrm{Del}(\sigma, \mathcal{F}_t^{\Updownarrow})$, so that $\Delta > \epsilon n$. Let $s = 20\sqrt{\frac{tn}{\epsilon}}$ and consider the random variable $Q = I_n^{\mathrm{Poi}}(s)$. We shall prove that $\Pr[\sigma_{|Q} \notin \mathcal{F}_t^{\Updownarrow}] \geq 5/6$.

Let $r = t - 1$. For $2 \leq u \leq t$, we define $\hat{\mathcal{F}}_u$ as follows.

$$\hat{\mathcal{F}}_u = \begin{cases} \mathcal{F}_u^{\Updownarrow} & t-u \text{ is even} \\ \mathcal{F}_u^{inv(\Updownarrow)} & t-u \text{ is odd} \end{cases}$$

We define a process that given a sample Q, tries to find evidence that $\sigma_{|Q}$ does not belong to $\mathcal{F}_t^{\Updownarrow}$. Following this, we analyze the probability that Q is such that the process succeeds.

To be precise, our process aims to find indices $a_0 < b_0 < a_1 < b_1 < \cdots < a_r < b_r$ and numbers $y_0, \ldots, y_r$, such that the following holds for every $0 \leq i \leq r$:

- **Property 1:** If i is odd, then (a_i, b_i) is a $\Updownarrow$-pair, and if i is even and $i > 0$, then (a_i, b_i) is an $\mathrm{inv}(\Updownarrow)$-pair.
- **Property 2:** $\mathrm{Del}(\sigma_{b_i+1} \ldots \sigma_n, \hat{\mathcal{F}}_{t-i}) \geq \Delta - 2\sum_{j=0}^{i} y_j$.

Notice that if the process succeeds, then in particular $((a_1, b_1), \ldots, (a_r, b_r))$ is an r-$\Updownarrow$-pair sequence, which implies by Observation 3.5 that $\sigma_{|Q} \notin \mathcal{F}_t^{\Updownarrow}$.

The process initializes $a_0 = -1$, $b_0 = 0$ and $y_0 = 0$, so that for $i = 0$ Property 1 holds trivially and Property 2 holds by Δ's definition.

For each $\ell \in [r]$, we henceforth condition on the process having succeeded in finding indices $a_0 < b_0 < \cdots < a_{\ell-1} < b_{\ell-1}$ and $y_0, \ldots, y_{\ell-1}$ such that Property 1 and Property 2 both hold for every $i \leq \ell - 1$. We now try to find a_ℓ, b_ℓ and y_ℓ such that $b_{\ell-1} < a_\ell < b_\ell$, and

both properties hold for $i = \ell$ as well. We refer to this attempt as the ℓ^{th} step of the process. For convenience, we assume that ℓ is odd (the other case is analogous). According to our assumption, since Property 2 holds for $i = \ell - 1$, we have that

$$\mathrm{Del}(\sigma_{b_{\ell-1}+1} \dots \sigma_n, \mathcal{F}^{\Updownarrow}_{t-\ell+1}) \geq \Delta - 2\sum_{j=0}^{\ell-1} y_j \ . \tag{3}$$

By Claim 5.4, there exist at least $\frac{1}{2}\mathrm{Del}(\sigma_{b_{\ell-1}+1} \dots \sigma_n, \mathcal{F}^{\Updownarrow}_{t-\ell+1})$ disjoint $\Updownarrow$-pairs in $\sigma_{b_{\ell-1}+1} \dots \sigma_n$, which by Equation (3) is at least $\frac{1}{2}\Delta - \sum_{j=0}^{\ell-1} y_j$. Consider the first $\Updownarrow$-pair, second $\Updownarrow$-pair, etc. in $\sigma_{b_{\ell-1}+1}, \dots, \sigma_n$, as defined in Definition 5.3. Assume that the first $\Updownarrow$-pair among them that belongs to Q is the q^{th} $\Updownarrow$-pair. Then we set $y_\ell = q$, a_ℓ to be the first index of this pair, and b_ℓ to be the last index of this pair (if no such pair belongs to Q, set $y_j = a_j = b_j = \infty$ for every $\ell \leq j \leq r$). If $y_\ell < \frac{1}{2}\Delta - \sum_{j=0}^{\ell-1} y_j$, then by using Claim 5.5 with $y = y_\ell$ and $m = b_\ell$, we infer that $\mathrm{Del}(\sigma_{b_\ell+1} \dots \sigma_n, \mathcal{F}^{\mathrm{inv}(\Updownarrow)}_{r-\ell}) \geq \Delta - 2\sum_{j=0}^{\ell} y_j$ (notice that the last sum now includes y_ℓ).

We conclude that if

$$y_\ell < \frac{1}{2}\Delta - \sum_{j=0}^{\ell-1} y_j \text{ for every } \ell \in [r] \ , \tag{4}$$

then the process succeeds and thus $\sigma_{|Q} \notin \mathcal{F}^{\Updownarrow}_t$. As $y_\ell \geq 0$ for every $\ell \in [r]$ and as $y_0 = 0$, the condition in Equation (4) is equivalent to

$$\sum_{j=1}^{r} y_j < \frac{1}{2}\Delta \ . \tag{5}$$

We now turn to analyze the probability that the condition in Equation (5) holds. Notice that by the definition of $Q = I_n^{\mathrm{Poi}}(s)$, each one of the pairs that were considered during the ℓ^{th} step of the process was sampled with probability ρ^2 for $\rho = 1 - e^{-\frac{s}{n}}$, independent of the others. Observe that from its construction, y_ℓ (for every $\ell \in [r]$) is distributed very similarly to a geometric random variable with parameter ρ^2. To be precise, for every $\ell \in [r]$ let z_ℓ be an i.i.d. geometric random variable with parameter ρ^2, and define $h_\ell = z_\ell$ if z_ℓ is no bigger than the number of pairs that were considered during the ℓ^{th} step, and $h_\ell = \infty$ otherwise. Then h_ℓ and y_ℓ have the same probability distribution. Therefore,

$$\begin{aligned}
\mathrm{Pr}_Q[\sigma_{|Q} \notin \mathcal{F}^{\Updownarrow}_t] \ &\geq \ \mathrm{Pr}_Q\left[\sum_{j=1}^{r} y_j < \frac{1}{2}\Delta\right] \ = \ \mathrm{Pr}_{Q,\{z_\ell\}_{\ell\in[r]}}\left[\sum_{j=1}^{r} h_j \ < \ \frac{1}{2}\Delta\right] \\
&\geq \ \mathrm{Pr}_{\{z_\ell\}_{\ell\in[r]}}\left[\sum_{j=1}^{r} z_j < \frac{1}{2}\Delta\right] \ \geq \ 1 - \mathrm{Pr}_{\{z_\ell\}_{\ell\in[r]}}\left[\sum_{j=1}^{r} z_j > \frac{1}{2}\epsilon n\right] \ .
\end{aligned}$$

Since $z_\ell \sim G(\rho^2)$ for every $\ell \in [r]$ and they are mutually independent, we know by Fact A.4 that their sum distributes as a negative binomial random variable, and that $\mathrm{Pr}_{\{z_\ell\}_{\ell\in[r]}}[\sum_{j=1}^{r} z_j > \frac{1}{2}\epsilon n] = \Pr[\mathrm{Bin}(\frac{1}{2}\epsilon n, \rho^2) < r]$. Let $\mu = \mathbb{E}[\mathrm{Bin}(\frac{1}{2}\epsilon n, \rho^2)] = \frac{1}{2}\epsilon n \rho^2$, and recall that $\rho = 1 - e^{-\frac{s}{n}}$. Applying Fact A.1 we get $\rho \geq \frac{s}{2n}$, so $\mu \geq \frac{\epsilon s^2}{8n}$. For $s = 20\sqrt{\frac{tn}{\epsilon}}$, the expected value of μ is at least $50t$, which is at least $10r$. Using a tail bound for the binomial distribution (Fact A.3) we conclude that

$$\Pr\left[\frac{1}{2}\mathrm{Bin}(\epsilon n, \rho) < r\right] \leq \Pr\left[\frac{1}{2}\mathrm{Bin}(\epsilon n, \rho) < \frac{1}{10}\mu\right] \leq e^{-\frac{(\frac{9}{10})^2\mu}{2}} \leq e^{-\frac{2}{5}\mu} \leq e^{-\frac{2}{5}\cdot 50t} \leq e^{-20} < \frac{1}{6} \ ,$$

which means that $\mathrm{Pr}_Q\left[\sigma_{|Q} \notin \mathcal{F}^{\Updownarrow}_t\right] \geq 5/6$, as claimed. ◀

The upper bound in Theorem 1.3 follows by combining Observation 3.7 with Lemma 5.6 and Observation 3.3 (in an analogous fashion to what was shown in the proof of Theorem 1.1).

References

1 Nir Ailon and Bernard Chazelle. Information theory in property testing and monotonicity testing in higher dimensions. *Information and Computation*, 204(11):1704–1717, 2006.

2 Nir Ailon, Bernard Chazelle, Ding Liu, and C. Seshadhri. Estimating the distance to a monotone function. *Random Structures and Algorithms*, 31(3):371–383, 2007.

3 Maria Balcan, Eric Blais, Avrim Blum, and Liu Yang. Active property testing. In *Proceedings of the Fifty-Third Annual Symposium on Foundations of Computer Science (FOCS)*, pages 21–30, 2012.

4 Roksana Baleshzar, Deeparnab Chakrabarty, Ramesh Krishnan, S. Pallavoor, and Sofya Raskhodnikova. Optimal unateness testers for real-valued functions. In *Automata, Languages and Programming: Forty-fourth International Colloquium (ICALP)*, pages 5:1–5:14, 2017.

5 Tugkan Batu, Ravi Kumar, and Ronitt Rubinfeld. Sublinear algorithms for testing monotone and unimodal distributions. In *Proceedings of the Thirty-Sixth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 381–390, 2004.

6 Tugkan Batu, Ronitt Rubinfeld, and Patrick White. Fast approximate PCPs for multidimensional bin-packing problems. *Information and Computation*, 196(1):42–56, 2005.

7 Aleksandra Belovs and Eric Blais. A polynomial lower bound for testing monotonicity. In *Proceedings of the Fourty-Eighth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 1021–1032, 2016.

8 Omri Ben-Eliezer. Testing local properties of arrays. In *Proceedings of the Tenth Innovations in Theoretical Computer Science conference (ITCS)*, pages 11:1–11:20, 2019.

9 Omri Ben-Eliezer and Clément L. Canonne. Improved bounds for testing forbidden order patterns. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2093–2112, 2018.

10 Omri Ben-Eliezer, Clément L. Canonne, Shoham Letzter, and Erik Waingarten. Finding monotone patterns in sublinear time. In *Proceedings of the Sixteeth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1469–1494, 2019.

11 Piotr Berman, Meiram Murzabulatov, and Sofya Raskhodnikova. Testing convexity of figures under the uniform distribution. In *Proceedings of the Thirty-Second International Symposium on Computational Geometry (SoCG)*, pages 17:1–17:15, 2016.

12 Arnab Bhattacharyya, Elena Grigorescu, Kyomin Jung, Sofya Raskhodnikova, and David P. Woodruff. Transitive-closure spanners. *SIAM Journal on Computing*, 41(6):1380–1425, 2012.

13 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 of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2133–2151, 2018.

14 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 Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1975–1994, 2020.

15 Eric Blais, Sofya Raskhodnikova, and Grigory Yaroslavtsev. Lower bounds for testing properties of functions over hypergrid domains. In *Proceedings of the Twenty-Ninth IEEE Annual Conference on Computational Complexity (CCC)*, pages 309–320, 2014.

16 Jop Briet, Sourav Chakraborty, David García-Soriano, and Arie Matsliah. Monotonicity testing and shortest-path routing on the cube. *Combinatorica*, 32(1):35–53, 2012.

17 Clément L. Canonne, Elena Grigorescu, Siyao Guo, Akash Kumar, and Karl Wimmer. Testing k-monotonicity. In *Proceedings of the Eighth Innovations in Theoretical Computer Science conference (ITCS)*, pages 29:1–29:21, 2017.

18 Deeparnab Chakrabarty, Kashyap Dixit, Madhav Jha, and C. Seshadhri. Property testing on product distributions: Optimal testers for bounded derivative properties. *ACM Transactions on Algorithms*, 13(2):20:1–20:30, 2017.

19 Deeparnab Chakrabarty and C. Seshadhri. Optimal bounds for monotonicity and Lipschitz testing over hypercubes and hypergrids. In *Proceedings of the Fourty-Fifth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 419–428, 2013.

20 Deeparnab Chakrabarty and C. Seshadhri. An optimal lower bound for monotonicity testing over hypergrids. *Theory of Computing*, 10:453–464, 2014.

21 Deeparnab Chakrabarty and C. Seshadhri. An $o(n)$ monotonicity tester for Boolean functions over the hypercube. *SIAM Journal on Computing*, 45(2):461–472, 2016.

22 Deeparnab Chakrabarty and C. Seshadhri. Adaptive Boolean monotonicity testing in totla influence time. In *Proceedings of the Tenth Innovations in Theoretical Computer Science conference (ITCS)*, pages 20:1–20:7, 2019.

23 Xi Chen, Anindya De, Rocco A. Servedio, and Li-Yang Tan. Boolean function monotonicity testing requires (almost) $n^{1/2}$ non-adaptive queries. In *Proceedings of the Fourty-Seventh Annual ACM Symposium on the Theory of Computing (STOC)*, pages 519–528, 2015.

24 Xi Chen, Rocco A. Servedio, and Li-Yang Tan. New algorithms and lower bounds for monotonicity testing. In *Proceedings of the Fifty-Fifth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 286–295, 2014.

25 Xi Chen and Erik Waingarten. Testing unateness nearly optimally. In *Proceedings of the Fifteeth-First Annual ACM Symposium on the Theory of Computing (STOC)*, pages 547–558, 2019.

26 Xi Chen, Erik Waingarten, and Jinyu Xie. Beyond Talagrand functions: New lower bounds for testing monotonicity and unateness. In *Proceedings of the Fourty-Ninth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 523–536, 2017.

27 Xi Chen, Erik Waingarten, and Jinyu Xie. Boolean unateness testing with $\tilde{O}(n^{3/4})$ adaptive queries. In *Proceedings of the Fifty-Eighth Annual Symposium on Foundations of Computer Science (FOCS)*, pages 868–879, 2017.

28 Ilias Diakonikolas and Daniel Kane. A new approach for testing properties of discrete distributions. In *Proceedings of the Fifty-Seventh Annual Symposium on Foundations of Computer Science (FOCS)*, pages 685–694, 2016. doi:10.1109/FOCS.2016.78.

29 Yevgeniy Dodis, Oded Goldreich, Eric Lehman, Sofya Raskhodnikova, Dana Ron, and Alex Samorodnitsky. Improved bounds for testing monotonicity. In *Proceedings of the Third International Workshop on Randomization and Approximation Techniques in Computer Science (RANDOM)*, pages 97–108, 1999.

30 Funda Ergün, Sampath Kannan, Ravi Kumar, Ronitt Rubinfeld, and Mahesh Viswanathan. Spot-checkers. In *Proceedings of the Thirtieth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 259–268, January 1998.

31 Shahar Fattal and Dana Ron. Approximating the distance to monotonicity in high dimensions. *ACM Transactions on Algorithms*, 6(3):52:1–52:37, 2010.

32 Eldar Fischer. On the strength of comparisons in property testing. *Information and Computation*, 189(1):107–116, 2004.

33 Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings of the Thirty-Fourth Annual ACM Symposium on the Theory of Computing (STOC)*, pages 474–483, 2002.

34 Oded Goldreich. The uniform distribution is complete with respect to testing identity to a fixed distribution. ECCC TR16-015, 2016.

35 Oded Goldreich, Shafi Goldwasser, Eric Lehman, Dana Ron, and Alex Samorodnitsky. Testing monotonicity. *Combinatorica*, 20(3):301–337, 2000.

36 Oded Goldreich and Dana Ron. On sample-based testers. *ACM Transactions on Computing Theory*, 8(2):7:1–7:54, 2016.

37 Shirley Halevy and Eyal Kushilevitz. Distribution-free property testing. *SIAM Journal on Computing*, 37(4):1107–1138, 2007.

38 Shirley Halevy and Eyal Kushilevitz. Testing monotonicity over graph products. *Random Structures and Algorithms*, 33(1):44–67, 2008.

39 Michael Kearns and Dana Ron. Testing problems with sub-learning sample complexity. *Journal of Computer and System Sciences*, 61(3):428–456, 2000.

40 Subhash Khot, Dor Minzer, and Muli Safra. On monotonicity testing and Boolean isoperimentric-type theorems. *SIAM Journal on Computing*, 47(6):2238–2276, 2018.

41 Subhash Khot and Igor Shinkar. An $\tilde{O}(n)$ queries adaptive tester for unateness. In *Proceedings of the Twentieth International Workshop on Randomization and Computation (RANDOM)*, pages 37:1–37:7, 2016.

42 Ramesh Krishnan, S. Pallavoor, Sofya Raskhodnikova, and Nithin M. Varma. Paramaterized property testing of functions. *ACM Transactions on Computing Theory*, 9(4):17:1–17:19, 2018.

43 Ramesh Krishnan, S. Pallavoor, Sofya Raskhodnikova, and Erik Waingarten. Approximating the distance to monotonicity of Boolean functions. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1995–2009, 2020.

44 Ilan Newman, Yuri Rabinovich, Deepak Rajendraprasad, and Christian Sohler. Testing for forbidden order patterns in an array. *Random Structures and Algorithms*, 55(2):402–426, 2019.

45 Michal Parnas, Dana Ron, and Ronitt Rubinfeld. Tolerant property testing and distance approximation. *Journal of Computer and System Sciences*, 72(6):1012–1042, 2006.

46 Sofya Raskhodnikova. Monotonicity testing. Master's thesis, MIT, 1999. Master's thesis.

47 Sofya Raskhodnikova, Dana Ron, Amir Shpilka, and Adam Smith. Strong lower bounds for approximating distribution support size and the distinct elements problem. *SIAM Journal on Computing*, 39(3):813–842, 2009.

48 Wojciech Szpankowski. *Average Case Analysis of Algorithms on Sequences*. Wiley-Interscience, New York, 2001.

A Basic facts and claims

▶ **Fact A.1** (Exponential inequality). *For $0 \leq p \leq 1$,*

$$\frac{1}{2}p \leq p - \frac{1}{2}p^2 \leq 1 - e^{-p} \leq p \ .$$

▶ **Fact A.2** (Markov's inequality). *If X is a nonnegative random variable and $a > 0$, then*

$$\Pr[X \geq a] \leq \frac{\mathbb{E}[x]}{a} \ .$$

▶ **Fact A.3** (Tail bound for the binomial distribution). *Let μ denote the expected value of $\mathrm{Bin}(n,p)$, i.e., $\mu = np$. Then for $\delta \in [0,1]$,*

$$\Pr[\mathrm{Bin}(n,p) \leq (1-\delta)\mu] \leq e^{-\frac{\delta^2\mu}{2}} \ ,$$

▶ **Fact A.4** (Sums of independent geometrically distributed random variables). *Let $W(n,p)$ denote the sum of n independent geometric random variables with parameter p. The variable $W(n,p)$ is said to have* negative binomial distribution*, and it satisfies*

$$\Pr[W(n,p) \leq m] = \Pr[\mathrm{Bin}(m,p) \geq k] \ .$$

B Proof of Observation 3.6

Since $\mathcal{M}_k$ is a hereditary property, $\mathrm{del}(\sigma, \mathcal{M}_k, p) \leq \mathrm{dist}(\sigma, \mathcal{M}_k, p)$. We next show that $\mathrm{dist}(\sigma, \mathcal{M}_k, p) \leq \mathrm{del}(\sigma, \mathcal{M}_k, p)$. By the definition of $\mathrm{del}(\sigma, \mathcal{M}_k, p)$, there exists a subset $D \subseteq [n]$ such that $\sigma_{[n]\setminus D} \in \mathcal{M}_k$ and $p(D) = \mathrm{del}(\sigma, \mathcal{M}_k, p)$. If there is more than one such subset, we select one with minimal size, so that $D \neq [n]$. We next define a sequence τ such that $\tau \in \mathcal{M}_k$, and such that $\tau_i = \sigma_i$ for every $i \in [n] \setminus D$, implying that $\mathrm{dist}(\sigma, \mathcal{M}_k, p) \leq \mathrm{dist}(\sigma, \tau, p) = p(D) \leq \mathrm{del}(\sigma, \mathcal{M}_k, p)$, as desired. We define τ as follows: For each $i \in [n]$, if $i \notin D$, then $\tau_i = \sigma_i$, and if $i \in D$, then $\tau_i = \sigma_j$ for $j \notin D$ such that $|j - i|$ is minimized (breaking ties arbitrarily). To verify that $\tau \in \mathcal{M}_k$, assume, contrary to the claim, that $\tau \notin \mathcal{M}_k$. By Observation 3.3, this means that for $\Updownarrow \in \{\Uparrow, \Downarrow\}$ and $t = k + 3$, there is a t-$\Updownarrow$ subsequence $(x_1, \ldots, x_t)$ with respect to τ. But by the definition of τ, this implies that there exists a t-$\Updownarrow$ subsequence $(x'_1, \ldots, x'_t)$ with respect to $\sigma_{[n]\setminus D}$, and we have reached a contradiction.

C The lower bounds

In this section we prove Theorem 1.3 and the lower bound of Theorem 1.2. Given n, k and ϵ, we construct a sequence and a corresponding probability distribution that are determined by a parameter m. The two lower bounds differ in the setting of this parameter.

Let ϵ, k satisfy $\epsilon < 1/4$ and $k < n/4 - 1$, and let m be an integer satisfying $2k \leq m < n/2$. Consider the sequence $\sigma = 2, 1, 4, 3, \ldots, 2m, 2m-1, 3m, 3m, \ldots, 3m$ (where the value $3m$ appears $n - 2m$ times and was chosen as an arbitrary value greater than $2m$). Define $p : [n] \to [0, 1]$ by $p(i) = \rho = \frac{2\epsilon}{m}$ for $i \leq 2m$ and $p(i) = \frac{1-2m\rho}{n-2m} = \frac{1-4\epsilon}{n-2m}$ for $i > 2m$, so that $\sum_{i\in[n]} p(i) = 1$. We shall show that σ is ϵ-far from being k-modal with respect to p, but the probability that a sample of size $s = \frac{1}{5}\frac{\sqrt{km}}{\epsilon}$, selected according to p, contains a subsequence that is not k-modal, is a small constant.

First, note that σ has exactly m descents: $(1, 2), \ldots, (2m-1, 2m)$, that is: $(2i-1, 2i)$ for every $i \in [m]$. Next, observe that any subsequence of σ with k descents is not k-modal (in fact, $\frac{k}{2} + 2$ descents are sufficient). Thus, by deleting at most $m - k$ indices, it is possible to eliminate at most $m - k$ descents, so that the resulting sequence is not k-modal. Since $p(i) = \rho$ for every $i \in [2m]$, we get that $\mathrm{del}(\sigma, \mathcal{F}_k, p) > (m - k)\rho \geq (m/2)\rho = \epsilon$, where the last inequality follows from the condition $m \geq 2k$. By Observation 3.6, $\mathrm{dist}(\sigma, \mathcal{F}_k, p) > \epsilon$, as claimed.

We now turn to show that the probability of sampling a subsequence of σ that is not k-modal is very low using $s = \frac{1}{5}\frac{\sqrt{km}}{\epsilon}$ samples. We do so by bounding the number of descents in the sampled subsequence, and the proof is a variant of a birthday-paradox argument. Let $q_1, \ldots, q_s$ denote our s samples. For every two different indices $\alpha, \beta \in [s]$, we define the event $E_{\alpha,\beta} = \{(q_\alpha, q_\beta) \text{ is a descent}\}$. Since there are exactly m descents and they are all disjoint, there are m options (each of weight ρ) for a first index in a descent, and given such an index there is exactly one option for the second index. As any two samples are independent, this means that $\Pr[E_{\alpha,\beta}] = m\rho \cdot \rho = m\rho^2$ for every $\alpha, \beta \in [s]$ such that $\alpha \neq \beta$. For every $\alpha, \beta \in [s]$ such that $\alpha \neq \beta$, let $\chi_{\alpha,\beta} = \mathbb{1}_{E_{\alpha,\beta}}$ denote the indicator function of the event $E_{\alpha,\beta}$. Then $X = \sum_{\alpha\neq\beta} \chi_{\alpha,\beta}$ is the number of descents in our sample. Using linearity of expectation, we can calculate its expected value:

$$\mathbb{E}[X] = \sum_{\alpha\neq\beta} \mathbb{E}[\chi_{\alpha,\beta}] = \sum_{\alpha\neq\beta} \Pr[E_{\alpha,\beta}] = s(s-1) \cdot m\rho^2 < \frac{km}{25\epsilon^2} \cdot m \cdot \left(\frac{2\epsilon}{m}\right)^2 < \frac{1}{6}k \,.$$

By Markov's inequality (Fact A.2),

$$\Pr\left[X \geq \frac{1}{2}k\right] \leq \Pr\left[X \geq 3\mathbb{E}[x]\right] \leq \frac{1}{3}\,.$$

Therefore with probability at least 2/3 the sampled subsequence contains less than $k/2$ descents and thus must be k-modal. Hence the tester rejects with a small constant probability.

We next choose appropriate values of m and infer Theorem 1.3 and the lower bound of Theorem 1.2.

Proof of Theorem 1.2. Set $m = \lfloor (n-1)/2 \rfloor$, and note that m fulfills its requirements as $k \leq n/4 - 1$ by the premise of the theorem. We conclude that $\Omega(\frac{\sqrt{km}}{\epsilon}) = \Omega(\frac{\sqrt{kn}}{\epsilon})$ samples are necessary for distribution-free one-sided error sample-based testing of k-modality. ◀

Proof of Theorem 1.3, lower bound. We shall assume that $2\epsilon n$ is an integer (otherwise, the analysis is similar but more cumbersome). Set $m = 2\epsilon n$, and note that m fulfills its requirements by the premise of the theorem that $k \leq \epsilon n$ and $\epsilon < 1/4$. We get that both $\rho = 2\epsilon/m = 1/n$ and $\frac{1-4\epsilon}{n-2m} = 1/n$, hence p is the uniform distribution over n. We conclude that $\Omega(\frac{\sqrt{km}}{\epsilon}) = \Omega(\sqrt{\frac{kn}{\epsilon}})$ samples are necessary for one-sided error sample-based testing of k-modality under the uniform distribution. ◀

When Is Amplification Necessary for Composition in Randomized Query Complexity?

Shalev Ben-David
University of Waterloo, Canada
shalev.b@uwaterloo.ca

Mika Göös
Stanford University, CA, USA
goos@stanford.edu

Robin Kothari
Microsoft Quantum and Microsoft Research, Redmond, WA, USA
robin.kothari@microsoft.com

Thomas Watson
University of Memphis, TN, USA
Thomas.Watson@memphis.edu

Abstract

Suppose we have randomized decision trees for an outer function f and an inner function g. The natural approach for obtaining a randomized decision tree for the composed function $(f \circ g^n)(x^1, \ldots, x^n) = f(g(x^1), \ldots, g(x^n))$ involves amplifying the success probability of the decision tree for g, so that a union bound can be used to bound the error probability over all the coordinates. The amplification introduces a logarithmic factor cost overhead. We study the question: When is this log factor necessary? We show that when the outer function is parity or majority, the log factor can be necessary, even for models that are more powerful than plain randomized decision trees. Our results are related to, but qualitatively strengthen in various ways, known results about decision trees with noisy inputs.

2012 ACM Subject Classification Theory of computation → Oracles and decision trees

Keywords and phrases Amplification, composition, query complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.28

Category RANDOM

Funding *Thomas Watson*: Supported by NSF grant CCF-1657377.

Acknowledgements We thank Badih Ghazi for interesting discussions about this work, and we thank anonymous reviewers for their comments.

1 Introduction

A deterministic decision tree for computing a partial function $f\colon \{0,1\}^n \to Z$ is a binary tree where each internal node is labeled with an index from $[n]$ and each leaf is labeled with an output value from Z. On input $x \in \{0,1\}^n$, the computation follows a root-to-leaf path where at a node labeled with index i, the value of x_i is queried and the path goes to the left child if $x_i = 0$ and to the right child if $x_i = 1$. The leaf reached on input x must be labeled with the value $f(x)$ (if the latter is defined). The cost of the decision tree is its depth, i.e., the maximum number of queries it makes over all inputs. The deterministic query complexity of f is the minimum cost of any deterministic decision tree that computes f. We will consider several more general models of decision trees (randomized, etc.), so we repurpose traditional complexity class notation to refer to the various associated query complexity measures. Since


© Shalev Ben-David, Mika Göös, Robin Kothari, and Thomas Watson;
licensed under Creative Commons License CC-BY



P is the traditional complexity class corresponding to deterministic computation, we let $\mathsf{P}(f)$ denote the deterministic query complexity of f. (Some of the recent literature uses the notation $\mathsf{P}^{\mathsf{dt}}(f)$, but this paper deals exclusively with decision trees, so we drop the dt superscript.)

A randomized decision tree is a probability distribution over deterministic decision trees. Computing f with error ε means that for every input x (for which $f(x)$ is defined), the probability that the output is not $f(x)$ is at most ε. The cost of a randomized decision tree is the maximum depth of all the deterministic trees in its support. The randomized query complexity $\mathsf{BPP}_\varepsilon(f)$ is the minimum cost of any randomized decision tree that computes f with error ε. When we write $\mathsf{BPP}(f)$ with no ε specified, we mean $\varepsilon = 1/3$. A basic fact about randomized computation is that the success probability can be amplified, with a multiplicative overhead in cost, by running several independent trials and taking the majority vote of the outputs: $\mathsf{BPP}_\varepsilon(f) \leq O(\mathsf{BPP}(f) \cdot \log(1/\varepsilon))$. See [9] for a survey of classic results on query complexity.

If $f\colon \{0,1\}^n \to Z$ and $g\colon \{0,1\}^m \to \{0,1\}$ are two partial functions, their composition is $f \circ g^n\colon (\{0,1\}^m)^n \to Z$ where $(f \circ g^n)(x^1, \ldots, x^n) \coloneqq f(g(x^1), \ldots, g(x^n))$ (which is defined iff $g(x^i)$ is defined for all i and $f(g(x^1), \ldots, g(x^n))$ is defined). How does the randomized query complexity of $f \circ g^n$ depend on the randomized query complexities of f and g? A simple observation is that to design a randomized decision tree for $f \circ g^n$, we can take a 1/6-error randomized decision tree for f and replace each query – say to the i^{th} input bit of f – with a $1/6n$-error randomized decision tree for evaluating $g(x^i)$. By a union bound, with probability at least 5/6 all of the (at most n) evaluations of g return the correct answer, and so with probability at least 2/3 the final evaluation of f is also correct. Since $\mathsf{BPP}_{1/6n}(g) \leq O(\mathsf{BPP}_{1/n}(g))$, we can write this upper bound as

$$\mathsf{BPP}(f \circ g^n) \;\leq\; O(\mathsf{BPP}(f) \cdot \mathsf{BPP}_{1/n}(g)) \;\leq\; O(\mathsf{BPP}(f) \cdot \mathsf{BPP}(g) \cdot \log n). \tag{1}$$

When is this tight? It will take some effort to suitably formulate this question. We begin by reviewing known related results.

1.1 When is amplification necessary?

As for general lower bounds (that hold for all f and g), much work has gone into proving lower bounds on $\mathsf{BPP}(f \circ g^n)$ in terms of complexity measures of f and g that are defined using models more powerful than plain randomized query complexity [16, 3, 6, 4, 5]. In terms of just $\mathsf{BPP}(f)$ and $\mathsf{BPP}(g)$, the state-of-the-art is that $\mathsf{BPP}(f \circ g^n) \geq \Omega(\mathsf{BPP}(f) \cdot \sqrt{\mathsf{BPP}(g)})$ for all f and g [14]. Furthermore, it is known that the latter bound is sometimes tight: There exist partial boolean functions f and g such that $\mathsf{BPP}(f \circ g^n) \leq \widetilde{O}(\mathsf{BPP}(f) \cdot \sqrt{\mathsf{BPP}(g)})$ and $\mathsf{BPP}(f), \mathsf{BPP}(g) \geq \omega(1)$ [14, 5]. Thus (1) is far from being *always* tight, even without worrying about the need for amplification. However, it remains plausible that $\mathsf{BPP}(f \circ g^n) \geq \Omega(\mathsf{BPP}(f) \cdot \mathsf{BPP}(g))$ holds for all *total* f and all partial g. We take this as a working conjecture in this paper. This conjecture has been confirmed for some specific outer functions f, such as the identity function $\textsc{Id}\colon \{0,1\}^n \to \{0,1\}^n$ [21] (this is called a "direct sum" result) and the boolean functions $\textsc{Or}$, $\textsc{Xor}$ (parity), and $\textsc{Maj}$ (majority) [17]. These results, however, do not address the need for amplification in the upper bound (1). To formulate our question of whether (1) is tight, a first draft could be:

Question A, with respect to a particular f: Is (1) tight for *all* partial functions g?

This is not quite a fair question, for at least two reasons:

- Regarding the first inequality in (1): The simple upper bound actually shows $\mathsf{BPP}(f \circ g^n) \leq O(\mathsf{BPP}(f) \cdot \mathsf{BPP}_{1/\mathsf{BPP}(f)}(g))$ (the union bound is only over queries that take place, not over all possible queries). So for simplicity, let us restrict our attention to f satisfying $\mathsf{BPP}(f) \geq \Omega(n)$, which is the case for Id, Or, Xor, and Maj.
- Regarding the second inequality in (1): Some functions g satisfy $\mathsf{BPP}_{1/n}(g) \leq o(\mathsf{BPP}(g) \cdot \log n)$ (e.g., if $\mathsf{P}(g) \leq O(\mathsf{BPP}(g))$). So for simplicity, let us restrict our attention to g satisfying $\mathsf{BPP}_{1/n}(g) \geq \Omega(\mathsf{BPP}(g) \cdot \log n)$, which (as we show later) is the case for two partial functions GapOr and GapMaj defined as follows ($|x|$ denotes the Hamming weight of $x \in \{0,1\}^m$):

$$\textsc{GapOr}(x) \coloneqq \begin{cases} 0 & \text{if } |x| = 0 \\ 1 & \text{if } |x| = m/2 \end{cases} \quad \text{and} \quad \textsc{GapMaj}(x) \coloneqq \begin{cases} 0 & \text{if } |x| = m/3 \\ 1 & \text{if } |x| = 2m/3 \end{cases}.$$

Thus, a better formulation of Question A would be: Assuming $\mathsf{BPP}(f) \geq \Omega(n)$, is (1) tight for all partial g satisfying $\mathsf{BPP}_{1/n}(g) \geq \Omega(\mathsf{BPP}(g) \cdot \log n)$? Even with these caveats, the answer is always "no." It will be instructive to examine a counterexample. Let $\textsc{Which}\colon \{0,1\}^2 \to \{0,1\}$ be the partial function such that $\textsc{Which}(y)$ indicates the location of the unique 1 in y, under the promise that $|y| = 1$. Then $g = \textsc{Which} \circ \textsc{GapOr}^2$ takes an input of length $2m$ with the promise that there are exactly $m/2$ many 1s, either all in the left half or all in the right half, and outputs which half has the 1s. It turns out $\mathsf{BPP}(g) \leq O(1)$ and $\mathsf{BPP}_{1/n}(g) \geq \Omega(\log n)$ provided $m \geq \log n$ (for similar reasons as GapOr itself) and yet $\mathsf{BPP}(f \circ g^n) \leq O(\mathsf{BPP}(f))$ for all f: To compute $f \circ g^n$, we can run an optimal randomized decision tree for f and whenever it queries $g(x^i)$, we repeatedly query uniformly random bit positions of x^i until we find a 1 (so the value of $g(x^i)$ is determined by which half we found a 1 in). This has the same error probability as the randomized decision tree for f, and the total number of queries to the bits of $(x^1, \ldots, x^n)$ is $O(\mathsf{BPP}(f))$ in expectation, because for each i it takes $O(1)$ queries in expectation to locate a 1 in x^i. By Markov's inequality, with high constant probability this halts after only $O(\mathsf{BPP}(f))$ total queries. Thus by aborting the computation if it attempts to make too many queries, we obtain a randomized decision tree for $f \circ g^n$ that always makes $O(\mathsf{BPP}(f))$ queries, with only a small hit in the error probability.

Blais and Brody [7] adjust the statement of Question A so the answer becomes "yes" in the case $f = \textsc{Id}$. Specifically, they weaken the right-hand side in such a way that the above counterexample is ruled out. Defining[1] $\overline{\mathsf{BPP}}_\varepsilon(g)$ similarly to $\mathsf{BPP}_\varepsilon(g)$ but where the cost of a randomized decision tree is the maximum over all inputs (on which g is defined) of the expected number of queries, we now have $\overline{\mathsf{BPP}}_{1/n}(g) \leq \overline{\mathsf{BPP}}_0(g) \leq O(1)$ for the g from the counterexample. The theorem from [7] is $\mathsf{BPP}(f \circ g^n) \geq \Omega(\mathsf{BPP}(f) \cdot \overline{\mathsf{BPP}}_{1/n}(g))$ when $f = \textsc{Id}$, in other words, $\mathsf{BPP}(g^n) = \Omega(n \cdot \overline{\mathsf{BPP}}_{1/n}(g))$ (a "strong direct sum" result). [7] also explicitly asked whether similar results hold for other functions f. The corresponding conjecture for $f = \textsc{Or}$ is false (as we note below) while for $f = \textsc{Xor}$ and $f = \textsc{Maj}$ it remains open.

To make progress, we step back and ask a seemingly more innocuous version of the question:

Question B, with respect to a particular f: Is (1) tight for *some* partial function g?

It turns out the answer is "no" for $f = \textsc{Or}$ and is "yes" for both $f = \textsc{Xor}$ and $f = \textsc{Maj}$.

[1] [7] used the notation $\overline{\mathsf{R}}$ instead of $\overline{\mathsf{BPP}}$.

1.2 Decision trees with noisy inputs

Question B is related to "query complexity with noisy inputs" (introduced in [13]), so let us review the latter model: When input bit y_i is queried, the wrong bit value is returned to the decision tree with some probability $\leq 1/3$ (and the correct value of y_i is returned with the remaining probability). The "noise events" are independent across all queries, including multiple queries to the same input bit. Now the adversary gets to pick not only the input, but also the "noise probabilities." [13] distinguishes between two extreme possibilites: A static adversary has a single common noise probability for all queries, while a dynamic adversary can choose a different noise probability for each node in the decision tree. In this paper we make a reasonable compromise: The adversary gets to choose a tuple of noise probabilities $(\nu_1, \dots, \nu_n)$, and each query to y_i returns $1 - y_i$ with probability exactly ν_i. When a randomized decision tree computes f with error probability ε, that means for every input $y \in \{0,1\}^n$ and every noise probability tuple $(\nu_1, \dots, \nu_n)$ (with $\nu_i \leq 1/3$ for each i), the output is $f(y)$ with probability $\geq 1 - \varepsilon$ over the random noise and randomness of the decision tree. We invent the notation $\mathsf{BPP}^*(f)$ for the minimum cost of any randomized decision tree that computes f on noisy inputs, with error probability $1/3$. We have $\mathsf{BPP}^*(f) \leq O(\mathsf{BPP}(f) \cdot \log n) \leq O(n \log n)$ by repeating each query $O(\log n)$ times and taking the majority vote (to drive the noise probabilities down to $o(1/n)$), and using a union bound to absorb the noise probabilities into the error probability. The connection with composition is that $\mathsf{BPP}(f \circ g^n) \leq \mathsf{BPP}^*(f) \cdot \mathsf{BPP}(g)$, because to design a randomized decision tree for $f \circ g^n$, we can take a $1/3$-error randomized decision tree for f with noisy inputs, and replace each query – say to y_i – with a $1/3$-error randomized decision tree for evaluating $g(x^i)$.

There is a similar connection for 1-sided error and 1-sided noise. When a randomized decision tree has 1-sided error ε, that means on 0-inputs the output is wrong with probability 0, and on 1-inputs the output is wrong with probability at most ε. We let $\mathsf{RP}(g)$ denote the minimum cost of any randomized decision tree that computes g with 1-sided error $1/2$. Similarly, 1-sided noise means that when input bit y_i is queried, if the actual value is $y_i = 0$ then 1 is returned with probability 0, and if the actual value is $y_i = 1$ then 0 is returned with probability $\nu_i \leq 1/2$. We invent the notation $\mathsf{BPP}^\dagger(f)$ for the minimum cost of any randomized decision tree that computes f on 1-sided noisy inputs, with 2-sided error probability $1/3$. We have $\mathsf{BPP}(f) \leq \mathsf{BPP}^\dagger(f) \leq \mathsf{BPP}^*(f)$. The connection $\mathsf{BPP}(f \circ g^n) \leq \mathsf{BPP}^\dagger(f) \cdot \mathsf{RP}(g)$ holds like in the 2-sided noise setting. We officially record these observations:

▶ **Observation 1.** *For all f and g,*

$$\mathsf{BPP}(f \circ g^n) \;\leq\; \mathsf{BPP}^*(f) \cdot \mathsf{BPP}(g) \quad \textit{and} \quad \mathsf{BPP}(f \circ g^n) \;\leq\; \mathsf{BPP}^\dagger(f) \cdot \mathsf{RP}(g).$$

The upshot is that noisy upper bounds imply composition upper bounds, and composition lower bounds imply noisy lower bounds. There are many proofs of the result $\mathsf{BPP}^*(\textsc{Or}) \leq O(n)$ [13, 23, 25, 20]:

▶ **Theorem 2** ($\textsc{Or}$ never necessitates amplification)**.** $\mathsf{BPP}^*(O_R) \leq O(n)$ *and thus for every partial function g,*

$$\mathsf{BPP}(O_R \circ g^n) \;\leq\; O(n \cdot \mathsf{BPP}(g)).$$

Theorem 2 is not new, but in Appendix A we provide a particularly clean and elementary proof (related to, but more streamlined than, the proof in [23]). We mention that the proof straightforwardly generalizes to some other functions f, such as "odd-max-bit": $\textsc{Omb}(y) = 1$ iff the highest index of any 1 in y is odd.

We turn our attention to lower bounds. Various special-purpose techniques have been developed for proving query complexity lower bounds in the noisy setting [13, 12, 11, 20]. However, a conceptual consequence of Observation 1 is that special-purpose techniques are not generally necessary: We can just use techniques for lower bounding plain (non-noisy) randomized query complexity, applied to composed functions.

1.3 Lower bound for parity

[13] proved that $\mathsf{BPP}^*(\textsc{Xor})$ and $\mathsf{BPP}^*(\textsc{Maj})$ are $\Omega(n \log n)$. Although apparently not recorded in the literature, it is possible to generalize this result to show $\mathsf{BPP}^\dagger(\textsc{Xor})$ and $\mathsf{BPP}^\dagger(\textsc{Maj})$ are $\Omega(n \log n)$. However, we prove results even stronger than that, using the composition paradigm. Our results involve query complexity models that are more powerful than BPP, and even more powerful than the $\overline{\mathsf{BPP}}$ model from [7]. This follows a theme from a lot of prior work: Since BPP query complexity is rather subtle, we can make progress by studying related models that are somewhat more "well-behaved."

- As observed in [7], the $\overline{\mathsf{BPP}}$ model is equivalent to one where the cost is the worst-case (rather than expected) number of queries, and a randomized decision tree is allowed to abort (i.e., output a special symbol $\perp$) with at most a small constant probability, and the output should be correct with high probability conditioned on not aborting.
- If we strengthen the above model by allowing the non-abort probability to be arbitrarily close to 0 (rather than close to 1), but require that the non-abort probabilities are approximately the same for all inputs (within some factor close to 1), the resulting model has been called $\mathsf{2WAPP}$ ("2-sided weak almost-wide PP") [18, 17]. The "1-sided" version WAPP, defined later, will be relevant to us.
- If we further strengthen the model by allowing the non-abort probabilities to be completely unrelated for different inputs (and still arbitrarily close to 0), the resulting model has been called $\mathsf{PostBPP}$ ("BPP with post-selection") [18, 10].

We first consider the last of these models. $\mathsf{PostBPP}_\varepsilon(f)$ is the minimum cost of any randomized decision tree such that on every input x (for which $f(x)$ is defined), the probability of outputting $\perp$ is < 1, and the probability of outputting $f(x)$ is $\geq 1 - \varepsilon$ conditioned on not outputting $\perp$. Trivially, $\mathsf{PostBPP}(f) \leq \mathsf{BPP}(f)$. In fact, the $\mathsf{PostBPP}$ model is much more powerful than plain randomized query complexity; for example (noted in [18]) it can efficiently compute the aforementioned odd-max-bit function: $\mathsf{PostBPP}(\textsc{Omb}) \leq 1$.

For the noisy input setting, $\mathsf{PostBPP}^*$ and $\mathsf{PostBPP}^\dagger$ are defined in the natural way, and $\mathsf{PostBPP}(f \circ g^n) \leq \mathsf{PostBPP}^*(f) \cdot \mathsf{BPP}(g)$ and $\mathsf{PostBPP}(f \circ g^n) \leq \mathsf{PostBPP}^\dagger(f) \cdot \mathsf{RP}(g)$ hold like in Observation 1.

In Section 2 we prove something qualitatively much stronger than $\mathsf{BPP}^*(\textsc{Xor}) \geq \Omega(n \log n)$:

▶ **Theorem 3** ($\textsc{Xor}$ sometimes necessitates amplification). *For some partial function g, namely $g = \textsc{GapMaj}$ with $m \geq \log n$,*

$$\mathsf{PostBPP}(\textsc{Xor} \circ g^n) \;\geq\; \Omega(n \cdot \mathsf{BPP}_{1/n}(g)) \;\geq\; \Omega(n \log n \cdot \mathsf{BPP}(g)).$$

In particular, $\mathsf{PostBPP}^*(\textsc{Xor}) \geq \Omega(n \log n)$.

Let us compare Theorem 3 to two previous results.

- [12] proved that $\overline{\mathsf{BPP}}^*(\textsc{Xor}) \geq \Omega(n \log n)$ and that this lower bound holds even in the average-case setting (i.e., $\Omega(n \log n)$ queries are needed in expectation to succeed with high probability over a uniformly random input, random noise, and randomness of the decision tree). Our proof of Theorem 3 is simpler than the proof in [12] (though both proofs have a Fourier flavor), it also works in the average-case setting, and it yields a stronger result since the model is $\mathsf{PostBPP}$ instead of just $\overline{\mathsf{BPP}}$ (and the lower bound holds for composition rather than just noisy inputs). [11] presented a different simplified proof of the result from [12], but that proof does not generalize to $\mathsf{PostBPP}^*$.
- Our proof of Theorem 3 shows something analogous, but incomparable, to the strong direct sum from [7]. As we explain in Section 2, our proof shows that $\mathsf{PostBPP}(\textsc{Xor} \circ g^n) \geq \Omega(n \cdot \mathsf{PostBPP}_{1/n}(g))$ holds for *all* g (thus addressing a version of our Question A). Compared to the [7] result that $\overline{\mathsf{BPP}}(\textsc{Id} \circ g^n) \geq \Omega(n \cdot \overline{\mathsf{BPP}}_{1/n}(g))$ for all g, our result has the advantages of working for $f = \textsc{Xor}$ rather than $f = \textsc{Id}$ and yielding a qualitatively stronger lower bound ($\mathsf{PostBPP}$ rather than $\overline{\mathsf{BPP}}$ on the left side), but the disadvantage of also requiring the qualitatively stronger type of lower bound on g. Our result shows that if amplifying g requires a log factor in a very strong sense (even $\mathsf{PostBPP}$-type decision trees cannot avoid the log factor), then that log factor will be necessary when composing $\textsc{Xor}$ with g.

1.4 Lower bound for majority

Our main result strengthens the bound $\mathsf{BPP}^*(\textsc{Maj}) \geq \Omega(n \log n)$ from [13], mainly by holding for the stronger model WAPP (rather than just BPP), but also by directly handling 1-sided noise and by holding for composition rather than just noisy inputs.

$\mathsf{WAPP}_\varepsilon(f)$ is the minimum cost of any randomized decision tree such that for some $t > 0$, on input x the probability of outputting 1 is in the range $[(1-\varepsilon)t, t]$ if $f(x) = 1$, and in the range $[0, \varepsilon t]$ if $f(x) = 0$. The ε subscript should always be specified, because unlike BPP and $\mathsf{PostBPP}$, WAPP is not amenable to efficient amplification of the error parameter ε [18]. For every constant $0 < \varepsilon < 1/2$, we have $\mathsf{PostBPP}(f) \leq O(\mathsf{WAPP}_\varepsilon(f)) \leq O(\mathsf{BPP}(f))$.

WAPP-type query complexity has several aliases, such as "approximate conical junta degree" and "approximate query complexity in expectation," and it has recently played a central role in various randomized query (and communication) complexity lower bounds [22, 18, 16, 17]. One can think of WAPP as a nonnegative version of approximate polynomial degree (which corresponds to the class AWPP); in other words, it is a classical analogue of the polynomial method used to lower bound quantum algorithms.

For the noisy input setting, WAPP^* and $\mathsf{WAPP}^\dagger$ are defined in the natural way, and $\mathsf{WAPP}_\varepsilon(f \circ g^n) \leq \mathsf{WAPP}^*_\varepsilon(f) \cdot \mathsf{BPP}(g)$ and $\mathsf{WAPP}_\varepsilon(f \circ g^n) \leq \mathsf{WAPP}^\dagger_\varepsilon(f) \cdot \mathsf{RP}(g)$ hold like in Observation 1. We prove the following theorem, which shows that WAPP sometimes requires amplification, even in the one-sided noise setting.

▶ **Theorem 4** ($\textsc{Maj}$ sometimes necessitates amplification)**.** *For some partial function g, namely $g = \textsc{GapOr}$ with $m \geq \log n$, and some constant $\varepsilon > 0$,*

$$\mathsf{WAPP}_\varepsilon(\textit{Maj} \circ g^n) \;\geq\; \Omega(n \cdot \mathsf{BPP}_{1/n}(g)) \;\geq\; \Omega(n \log n \cdot \mathsf{RP}(g)).$$

In particular, $\mathsf{WAPP}^\dagger_\varepsilon(\textit{Maj}) \geq \Omega(n \log n)$.

This theorem should be contrasted with the work of Sherstov about making polynomials robust to noise [27]. In that work, Sherstov showed that approximate polynomial degree never requires a log factor in the noisy input setting, nor in composition. That is to say, he

improved the simple bound $\mathsf{AWPP}^*(f) \le O(\mathsf{AWPP}(f) \cdot \log n)$ to $\mathsf{AWPP}^*(f) \le O(\mathsf{AWPP}(f))$ for all Boolean functions f, and showed $\mathsf{AWPP}(f \circ g^n) \le O(\mathsf{AWPP}(f) \cdot \mathsf{AWPP}(g))$. In contrast, for conical juntas (nonnegative linear combinations of conjunctions), Theorem 4 shows that in a strong sense, the simple bound $\mathsf{WAPP}^*_\epsilon(f) \le O(\mathsf{WAPP}_\delta(f) \cdot \log n)$ (for all constants $0 < \delta < \varepsilon < 1/2$ and total Boolean functions f) cannot be improved: $\mathsf{WAPP}^\dagger_\varepsilon(f) \ge \Omega(\mathsf{WAPP}_0(f) \cdot \log n)$ for some constant ε and some total f, namely $f = \textsc{Maj}$. Thus unlike polynomials, conical juntas cannot be made robust to noise.

Our proof of Theorem 4 (in Section 3) introduces some technical ideas that may be useful for other randomized query complexity lower bounds.

By a simple reduction, Theorem 4 for $g = \textsc{GapOr}$ implies the same for $g = \textsc{GapMaj}$ (with $\mathsf{BPP}(g) = 1$ instead of $\mathsf{RP}(g) = 1$ at the end of the statement), but we do not know of a simpler direct proof for the latter result. Theorem 4 cannot be strengthened to have $\mathsf{PostBPP}$ in place of WAPP, because $\mathsf{PostBPP}(\textsc{Maj} \circ \textsc{GapMaj}^n) \le O(n)$. However, Theorem 4 does hold with $\textsc{Xor}$ in place of $\textsc{Maj}$, by the same proof.

2 Proof of Theorem 3: Xor sometimes necessitates amplification

We first discuss a standard technique for proving randomized query complexity lower bounds, which will be useful in the proof of Theorem 3. For any conjunction $C\colon \{0,1\}^k \to \{0,1\}$ and distribution $\mathcal{D}$ over $\{0,1\}^k$, we write $C(\mathcal{D}) \coloneqq \mathbb{E}_{x\sim\mathcal{D}}[C(x)] = \mathbb{P}_{x\sim\mathcal{D}}[C(x) = 1]$. The number of literals in a conjunction is called its width.

▶ **Fact 5.** *Let $h\colon \{0,1\}^k \to \{0,1\}$ be a partial function, and for each $z \in \{0,1\}$ let $\mathcal{D}_z$ be a distribution over $h^{-1}(z)$. Then for every ε there exist a conjunction C of width $\mathsf{PostBPP}_\varepsilon(h)$ and a $z \in \{0,1\}$ such that $\varepsilon \cdot C(\mathcal{D}_z) \ge (1-\varepsilon) \cdot C(\mathcal{D}_{1-z})$ and $C(\mathcal{D}_z) > 0$.*

Proof. Abbreviate $\mathsf{PostBPP}_\varepsilon(h)$ as r. Fix a randomized decision tree of cost r computing h with error ε conditioned on not aborting, and assume w.l.o.g. that for each outcome of the randomness, the corresponding deterministic tree is a perfect tree with 2^r leaves, all at depth r. Consider the probability space where we sample input x from the mixture $\frac{1}{2}\mathcal{D}_0 + \frac{1}{2}\mathcal{D}_1$, sample a deterministic decision tree T as an outcome of the randomized decision tree, and sample a uniformly random leaf ℓ of T. Let A be the indicator random variable for the event that ℓ is the leaf reached by $T(x)$ and its label is $h(x)$. Let B be the indicator random variable for the event that ℓ is the leaf reached by $T(x)$ and its label is $1 - h(x)$. Conditioned on any particular x and T, the probability that ℓ is the leaf reached by $T(x)$ is 2^{-r}. Thus conditioned on any particular x, if the non-abort probability is $t_x > 0$ then $\mathbb{E}[A \,|\, x] \ge 2^{-r} t_x (1-\varepsilon)$ and $\mathbb{E}[B \,|\, x] \le 2^{-r} t_x \varepsilon$ and thus $\varepsilon \cdot \mathbb{E}[A \,|\, x] - (1-\varepsilon) \cdot \mathbb{E}[B \,|\, x] \ge 0$. Over the whole probability space, we have $\varepsilon \cdot \mathbb{E}[A] - (1-\varepsilon) \cdot \mathbb{E}[B] \ge 0$, so by linearity the same must hold conditioned on some particular T and ℓ with $\mathbb{E}[A \,|\, T, \ell] > 0$. Let C be the conjunction of width r such that $C(x) = 1$ iff $T(x)$ reaches ℓ, and let z be the label of ℓ. Then we have $C(\mathcal{D}_z) = \mathbb{E}[A \,|\, T, \ell \text{ and } h(x) = z] = 2 \cdot \mathbb{E}[A \,|\, T, \ell] > 0$ and similarly $C(\mathcal{D}_{1-z}) = 2 \cdot \mathbb{E}[B \,|\, T, \ell]$. Thus

$$\varepsilon \cdot C(\mathcal{D}_z) - (1-\varepsilon) \cdot C(\mathcal{D}_{1-z}) \;=\; 2 \cdot \big(\varepsilon \cdot \mathbb{E}[A \,|\, T, \ell] + (1-\varepsilon) \cdot \mathbb{E}[B \,|\, T, \ell]\big) \;\ge\; 0. \qquad \blacktriangleleft$$

Now we work toward proving Theorem 3. Throughout, n is the input length of $\textsc{Xor}$, and m is the input length of $\textsc{GapMaj}$. We have $\mathsf{BPP}(\textsc{GapMaj}) \le 1$ by outputting the bit at a uniformly random position from the input. We describe one way of seeing that $\mathsf{BPP}_{1/n}(\textsc{GapMaj}) \ge \mathsf{PostBPP}_{1/n}(\textsc{GapMaj}) \ge \Omega(\log n)$ provided $m \ge \log n$. For $z \in \{0,1\}$, define $\mathcal{G}_z$ as the uniform distribution over $\textsc{GapMaj}^{-1}(z)$.

▶ **Fact 6.** *For every conjunction* $C\colon \{0,1\}^m \to \{0,1\}$ *of width* $w \le m/7$ *and for each* $z \in \{0,1\}$*,*

$$C(\mathcal{G}_z) \;\le\; 3^w \cdot C(\mathcal{G}_{1-z}).$$

Proof. By symmetry we just consider $z = 0$. Suppose C has u positive literals and v negative literals ($u + v = w$). Then

$$C(\mathcal{G}_0) \;=\; \tbinom{m-w}{m/3-u}/\tbinom{m}{m/3} \;\le\; \tbinom{m-w}{m/3}/\tbinom{m}{m/3} \;=\; \tfrac{(2m/3)\cdot(2m/3-1)\cdots(2m/3-w+1)}{m\cdot(m-1)\cdots(m-w+1)} \;\le\; (2/3)^w,$$

$$\begin{aligned} C(\mathcal{G}_1) \;=\; \tbinom{m-w}{m/3-v}/\tbinom{m}{m/3} \;\ge\; \tbinom{m-w}{m/3-w}/\tbinom{m}{m/3} \;&=\; \tfrac{(m/3)\cdot(m/3-1)\cdots(m/3-w+1)}{m\cdot(m-1)\cdots(m-w+1)} \\ &\ge\; \left(\tfrac{m/3-w}{m-w}\right)^w \;\ge\; \left(\tfrac{m/3-m/7}{m-m/7}\right)^w \;=\; (2/9)^w. \end{aligned}$$

Thus $C(\mathcal{G}_0)/C(\mathcal{G}_1) \le \left(\frac{2/3}{2/9}\right)^w = 3^w$. ◀

Combining Fact 5 and Fact 6 (using $h = \textsc{GapMaj}$, $k = m$, $\mathcal{D}_z = \mathcal{G}_z$, $\varepsilon = 1/n$, and $w = \mathsf{PostBPP}_\varepsilon(h)$) implies that $(1-\varepsilon)/\varepsilon \le 3^w$, in other words we have $\mathsf{PostBPP}_{1/n}(\textsc{GapMaj}) \ge \log_3(n(1-1/n)) \ge \Omega(\log n)$, provided $w \le m/7$. If $w > m/7$ then $\mathsf{PostBPP}_{1/n}(\textsc{GapMaj}) \ge \Omega(\log n)$ holds anyway provided $m \ge \log n$.

Hence, our result can be restated as follows.

▶ **Theorem 3 (Restated).** $\mathsf{PostBPP}(\textsc{Xor} \circ \textsc{GapMaj}^n) \ge \Omega(n \log n)$ *provided* $m \ge \log n$.

Proof. We show $\mathsf{PostBPP}(\textsc{Xor} \circ \textsc{GapMaj}^n) > \frac{1}{14} n \log n$. By Fact 5 (using $h = \textsc{Xor} \circ \textsc{GapMaj}^n$, $k = nm$, and $\varepsilon = 1/3$) it suffices to exhibit for each $z \in \{0,1\}$ a distribution $\mathcal{D}_z$ over $(\textsc{Xor} \circ \textsc{GapMaj}^n)^{-1}(z)$, such that for every conjunction C of width $\le \frac{1}{14} n \log n$ and for each $z \in \{0,1\}$, either $C(\mathcal{D}_z) < 2C(\mathcal{D}_{1-z})$ or $C(\mathcal{D}_z) = 0$. Letting $\mathcal{F}_z$ be the uniform distribution over $\textsc{Xor}^{-1}(z)$, define $\mathcal{D}_z$ as the mixture over $y \sim \mathcal{F}_z$ of $\mathcal{G}_y \coloneqq \mathcal{G}_{y_1} \times \cdots \times \mathcal{G}_{y_n}$ (i.e., $(x^1, \ldots, x^n) \sim \mathcal{G}_y$ is sampled by independently sampling $x^i \sim \mathcal{G}_{y_i}$ for all i). Put succinctly, $\mathcal{D}_z \coloneqq \mathbb{E}_{y \sim \mathcal{F}_z}[\mathcal{G}_y]$. Letting $\mathcal{G} \coloneqq \frac{1}{2}\mathcal{G}_0 + \frac{1}{2}\mathcal{G}_1$ and $\mathcal{F} \coloneqq \frac{1}{2}\mathcal{F}_0 + \frac{1}{2}\mathcal{F}_1$ and $\mathcal{D} \coloneqq \frac{1}{2}\mathcal{D}_0 + \frac{1}{2}\mathcal{D}_1$, we have $\mathcal{D} = \mathcal{G}^n$ since $\mathcal{F}$ is uniform over $\{0,1\}^n$. Since $C(\mathcal{D}) = \frac{1}{2}C(\mathcal{D}_0) + \frac{1}{2}C(\mathcal{D}_1)$, our goal of showing "$\frac{1}{2}C(\mathcal{D}_0) < C(\mathcal{D}_1) < 2C(\mathcal{D}_0)$ or $C(\mathcal{D}_0) = C(\mathcal{D}_1) = 0$" is equivalent to showing "$\frac{2}{3}C(\mathcal{D}) < C(\mathcal{D}_1) < \frac{4}{3}C(\mathcal{D})$ or $C(\mathcal{D}) = 0$."

Now consider any conjunction C of width $w \le \frac{1}{14} n \log n$ such that $C(\mathcal{D}) > 0$, and write $C(x^1, \ldots, x^n) = \prod_i C_i(x^i)$ where C_i is a conjunction. Since $C_i(\mathcal{G}) = \frac{1}{2}C_i(\mathcal{G}_0) + \frac{1}{2}C_i(\mathcal{G}_1)$, for each $y_i \in \{0,1\}$ we can write $C_i(\mathcal{G}_{y_i}) = (1 + a_i(-1)^{y_i})C_i(\mathcal{G})$ for some number a_i with $|a_i| \le 1$ (so $a_i \ge 0$ iff $C_i(\mathcal{G}_0) \ge C_i(\mathcal{G}_1)$). Let w_i be the width of C_i, so $\sum_i w_i = w \le \frac{1}{14} n \log n$. Then $w_i \le \frac{1}{7}\log n \le m/7$ for at least $n/2$ many values of i, and for such i note that by Fact 6, $C_i(\mathcal{G}_{y_i}) \le 3^{(\log n)/7} \cdot C_i(\mathcal{G}_{1-y_i}) \le n^{1/4} \cdot C_i(\mathcal{G}_{1-y_i})$ for each $y_i \in \{0,1\}$. The latter implies that $|a_i| \le 1 - 2/(n^{1/4}+1) \le 1 - n^{-1/4}$. Thus

$$\left|\textstyle\prod_i a_i\right| \;=\; \textstyle\prod_i |a_i| \;\le\; (1 - n^{-1/4})^{n/2} \;\le\; e^{-n^{3/4}/2} \;\le\; 1/4.$$

For $S \subseteq [n]$, let $\chi_S\colon \{0,1\}^n \to \{1,-1\}$ be the character $\chi_S(y) \coloneqq \prod_{i \in S}(-1)^{y_i} = (-1)^{\sum_{i \in S} y_i}$. Note that $\mathbb{E}_{y \sim \mathcal{F}_1}[\chi_S]$ is 1 if $S = \emptyset$, is -1 if $S = [n]$, and is 0 otherwise. Putting everything together,

$$\begin{aligned} C(\mathcal{D}_1) \;&=\; \mathbb{E}_{y \sim \mathcal{F}_1}[C(\mathcal{G}_y)] \;=\; \mathbb{E}_{y \sim \mathcal{F}_1}\left[\textstyle\prod_i C_i(\mathcal{G}_{y_i})\right] \;=\; \mathbb{E}_{y \sim \mathcal{F}_1}\left[\textstyle\prod_i (1 + a_i(-1)^{y_i})C_i(\mathcal{G})\right] \\ &=\; \left(\textstyle\prod_i C_i(\mathcal{G})\right) \cdot \mathbb{E}_{y \sim \mathcal{F}_1}\left[\textstyle\sum_{S \subseteq [n]} \prod_{i \in S} a_i(-1)^{y_i}\right] \\ &=\; C(\mathcal{D}) \cdot \textstyle\sum_{S \subseteq [n]} \left(\prod_{i \in S} a_i\right) \cdot \mathbb{E}_{y \sim \mathcal{F}_1}[\chi_S(y)] \\ &=\; C(\mathcal{D}) \cdot \left(1 - \textstyle\prod_{i \in [n]} a_i\right) \;\in\; C(\mathcal{D}) \cdot (1 \pm 1/4) \end{aligned}$$

which implies $\frac{2}{3}C(\mathcal{D}) < C(\mathcal{D}_1) < \frac{4}{3}C(\mathcal{D})$ since we are assuming $C(\mathcal{D}) > 0$. This concludes the proof of Theorem 3. ◀

Using strong LP duality (as in [15]), it can be seen that Fact 5 is a tight lower bound method up to constant factors: $\mathsf{PostBPP}_\varepsilon(h) \geq \Omega(c)$ iff it is possible to prove this via Fact 5 by exhibiting "hard input distributions" $\mathcal{D}_0$ and $\mathcal{D}_1$ (as we did for GapMaj in Fact 6). Since this was the only property of g used in the proof of Theorem 3, this implies that $\mathsf{BPP}(\textsc{Xor}\circ g^n) \geq \mathsf{PostBPP}(\textsc{Xor}\circ g^n) \geq \Omega(n\cdot\mathsf{PostBPP}_{1/n}(g))$ holds for all g, as we mentioned in Subsection 1.3.

3 Proof of Theorem 4: Maj sometimes necessitates amplification

We first discuss a standard technique for proving randomized query complexity lower bounds, which will be useful in the proof of Theorem 4. For any conjunction $C\colon \{0,1\}^k \to \{0,1\}$ and distribution $\mathcal{D}$ over $\{0,1\}^k$, we write $C(\mathcal{D}) \coloneqq \mathbb{E}_{x\sim\mathcal{D}}[C(x)] = \mathbb{P}_{x\sim\mathcal{D}}[C(x)=1]$. The number of literals in a conjunction is called its width.

▶ **Fact 7.** *Let $h\colon \{0,1\}^k \to \{0,1\}$ be a partial function, and let $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{D}_2$ be three distributions, over $h^{-1}(0)$, $h^{-1}(1)$, and $h^{-1}(0)\cup h^{-1}(1)$ respectively. Then for every $0 < \varepsilon \leq 1/10$ there exists a conjunction C of width $\mathsf{WAPP}_\varepsilon(h)$ such that $C(\mathcal{D}_0) \leq \delta\cdot C(\mathcal{D}_1)$ and $C(\mathcal{D}_2) \leq (1+\delta)\cdot C(\mathcal{D}_1)$ and $C(\mathcal{D}_1) > 0$, where $\delta \coloneqq 2\sqrt{\varepsilon}$.*

The key calculation underlying the proof of Fact 7 is encapsulated in the following:

▶ **Fact 8.** *Let P_0, P_1, P_2 be three jointly distributed nonnegative random variables with $\mathbb{E}[P_1] > 0$. For any $0 < \varepsilon \leq 1/10$, if $\mathbb{E}[P_0] \leq \varepsilon$ and $\mathbb{E}[P_1] \geq 1-\varepsilon$ and $\mathbb{E}[P_2] \leq 1$, then there exists an outcome o such that $P_0(o) \leq \delta\cdot P_1(o)$ and $P_2(o) \leq (1+\delta)\cdot P_1(o)$ and $P_1(o) > 0$, where $\delta \coloneqq 2\sqrt{\varepsilon}$.*

Proof of Fact 8. Let $W \coloneqq \{o : P_1(o) > 0\} \neq \emptyset$. Suppose for contradiction that for every outcome $o \in W$, either $P_0(o) > \delta\cdot P_1(o)$ or $P_2(o) > (1+\delta)\cdot P_1(o)$. Then W can be partitioned into events U and V such that $P_0(o) > \delta\cdot P_1(o)$ for every $o \in U$ and $P_2(o) > (1+\delta)\cdot P_1(o)$ for every $o \in V$. Letting I_U and I_V be the indicator random variables for these events, we have $\mathbb{E}[P_1\cdot I_U] + \mathbb{E}[P_1\cdot I_V] = \mathbb{E}[P_1]$ and thus either:

- $\mathbb{E}[P_1\cdot I_U] \geq \sqrt{\varepsilon}\cdot\mathbb{E}[P_1]$, in which case

$$\mathbb{E}[P_0] \;\geq\; \mathbb{E}[P_0\cdot I_U] \;>\; \delta\cdot\mathbb{E}[P_1\cdot I_U] \;\geq\; \delta\cdot\sqrt{\varepsilon}\cdot(1-\varepsilon) \;=\; 2\varepsilon(1-\varepsilon) \;>\; \varepsilon, \quad \text{or}$$

- $\mathbb{E}[P_1\cdot I_V] \geq (1-\sqrt{\varepsilon})\cdot\mathbb{E}[P_1]$, in which case

$$\mathbb{E}[P_2] \;\geq\; \mathbb{E}[P_2\cdot I_V] \;>\; (1+\delta)\cdot\mathbb{E}[P_1\cdot I_V] \;\geq\; (1+\delta)\cdot(1-\sqrt{\varepsilon})\cdot(1-\varepsilon) \;>\; 1$$

where the last inequality can be verified by a little calculus for $0 < \varepsilon \leq 1/10$.
Both cases yield a contradiction. ◀

Proof of Fact 7. Abbreviate $\mathsf{WAPP}_\varepsilon(h)$ as r. Fix a randomized decision tree of cost r computing h with error parameter ε and threshold $t > 0$ (from the definition of WAPP), and assume w.l.o.g. that for each outcome of the randomness, the corresponding deterministic tree is a perfect tree with 2^r leaves, all at depth r. Consider the probability space where we sample a deterministic decision tree T as an outcome of the randomized decision tree, and sample a uniformly random leaf ℓ of T. For any outcome T, ℓ, let $C_{T,\ell}$ be the conjunction of width r such that $C_{T,\ell}(x) = 1$ iff $T(x)$ reaches ℓ. Define three joint random variables P_0, P_1, P_2 as

$$P_j(T,\ell) \;\coloneqq\; \begin{cases} C_{T,\ell}(\mathcal{D}_j) & \text{if the label of } \ell \text{ is } 1 \\ 0 & \text{if the label of } \ell \text{ is } 0 \end{cases}.$$

Conditioned on any particular x and T, the probability that ℓ is the leaf reached by $T(x)$ is 2^{-r}. Thus

$$\begin{aligned}\mathbb{E}[P_j] &= \mathbb{P}_{T,\ell,\,x\sim\mathcal{D}_j}[\ell \text{ is the leaf reached by } T(x) \text{ and its label is } 1] \\ &= \mathbb{E}_{x\sim\mathcal{D}_j}\left[2^{-r}\cdot\mathbb{P}_T[T(x) \text{ outputs } 1]\right]\end{aligned}$$

which implies $\mathbb{E}[P_0] \le 2^{-r}t\varepsilon$ and $\mathbb{E}[P_1] \ge 2^{-r}t(1-\varepsilon)$ and $\mathbb{E}[P_2] \le 2^{-r}t$. Applying Fact 8 to the scaled random variables $(2^r/t)P_0$, $(2^r/t)P_1$, $(2^r/t)P_2$ yields an outcome T, ℓ such that

$$P_0(T,\ell) \le \delta\cdot P_1(T,\ell) \quad\text{and}\quad P_2(T,\ell) \le (1+\delta)\cdot P_1(T,\ell) \quad\text{and}\quad P_1(T,\ell) > 0.$$

Since $P_1(T,\ell) > 0$, the label of ℓ must be 1, so we get

$$C_{T,\ell}(\mathcal{D}_0) \le \delta\cdot C_{T,\ell}(\mathcal{D}_1) \quad\text{and}\quad C_{T,\ell}(\mathcal{D}_2) \le (1+\delta)\cdot C_{T,\ell}(\mathcal{D}_1) \quad\text{and}\quad C_{T,\ell}(\mathcal{D}_1) > 0.$$

◀

Now we work toward proving Theorem 4. Throughout, n is the input length of Maj, and m is the input length of GapOr. We have $\mathsf{RP}(\text{GapOr}) \le 1$ by outputting the bit at a uniformly random position from the input. We describe one way of seeing that $\mathsf{BPP}_{1/n}(\text{GapOr}) \ge \mathsf{WAPP}_{1/n}(\overline{\text{GapOr}}) \ge \Omega(\log n)$ provided $m \ge \log n$ (this cannot be shown via Fact 5). For $z \in \{0,1\}$, define $\mathcal{G}_z$ as the uniform distribution over $\text{GapOr}^{-1}(z)$.

▶ **Fact 9.** *For every conjunction $C\colon \{0,1\}^m \to \{0,1\}$:*
(i) *$C(\mathcal{G}_0) \in \{0,1\}$.*
(ii) *If $C(\mathcal{G}_0) = 1$ and C has width $w \le m/4$ then $C(\mathcal{G}_1) \ge 3^{-w}$.*

Proof. *(i)*: Note that $\mathcal{G}_0$ is supported entirely on the input 0^m. If C has a positive literal then $C(\mathcal{G}_0) = 0$. If C has only negative literals then $C(\mathcal{G}_0) = 1$.

(ii): Suppose C has w negative literals and no positive literals. Then

$$C(\mathcal{G}_1) = \tbinom{m-w}{m/2}/\tbinom{m}{m/2} = \tfrac{(m/2)\cdot(m/2-1)\cdots(m/2-w+1)}{m\cdot(m-1)\cdots(m-w+1)} \ge \left(\tfrac{m/2-w}{m-w}\right)^w \ge \left(\tfrac{m/2-m/4}{m-m/4}\right)^w = 3^{-w}.$$

◀

Combining Fact 7 and Fact 9 (using $h = \overline{\text{GapOr}}$, $k = m$, $\mathcal{D}_0 = \mathcal{G}_1$, $\mathcal{D}_1 = \mathcal{G}_0$, $\mathcal{D}_2$ is not needed, $\varepsilon = 1/n$, and $w = \mathsf{WAPP}_\varepsilon(h)$) implies that $3^{-w} \le \delta$, in other words $\mathsf{WAPP}_{1/n}(\overline{\text{GapOr}}) \ge \log_3(1/(2\sqrt{1/n})) \ge \Omega(\log n)$, provided $w \le m/4$. If $w > m/4$ then $\mathsf{WAPP}_{1/n}(\overline{\text{GapOr}}) \ge \Omega(\log n)$ holds anyway provided $m \ge \log n$.

Hence, our result can be restated as follows.[2]

▶ **Theorem 4 (Restated).** $\mathsf{WAPP}_\varepsilon(\text{Maj}\circ\text{GapOr}^n) \ge \Omega(n\log n)$ *for some constant $\varepsilon > 0$ provided $m \ge \log n$.*

We show $\mathsf{WAPP}_{1/36}(\text{Maj}\circ\text{GapOr}^n) > \frac{1}{16}n\log n$. By Fact 7 (using $h = \text{Maj}\circ\text{GapOr}^n$, $k = nm$, $\varepsilon = 1/36$, and $\delta = 1/3$) it suffices to exhibit distributions $\mathcal{D}_0$, $\mathcal{D}_1$, $\mathcal{D}_2$ over $h^{-1}(0)$, $h^{-1}(1)$, and $h^{-1}(0)\cup h^{-1}(1)$ respectively, such that for every conjunction C of width $\le \frac{1}{16}n\log n$, either $C(\mathcal{D}_0) > \frac{1}{3}C(\mathcal{D}_1)$ or $C(\mathcal{D}_2) > \frac{4}{3}C(\mathcal{D}_1)$ or $C(\mathcal{D}_1) = 0$. Assume n is

[2] Properties *(i)* and *(ii)* from Fact 9 are somewhat stronger than necessary for the proof of Theorem 4 to go through. The proof works, with virtually no modification, for any g satisfying the following for some distributions $\mathcal{G}_z$ over $g^{-1}(z)$ ($z \in \{0,1\}$): For every conjunction $C\colon \{0,1\}^m \to \{0,1\}$ such that $C(\mathcal{G}_0) > 0$, we have $C(\mathcal{G}_1) \le C(\mathcal{G}_0)$ and if furthermore C has width $w \le m/4$ then $C(\mathcal{G}_1) \ge 2^{-O(w)}\cdot C(\mathcal{G}_0)$.

even and for the tiebreaker, $\textsc{Maj}(y) = 1$ if $|y| = n/2$. For $\zeta \in \{0,1,2\}$ letting $\mathcal{F}_\zeta$ be the uniform distribution over all $y \in \{0,1\}^n$ with $|y| = n/2 - 1 + \zeta$ (so $\mathcal{F}_0$, $\mathcal{F}_1$, $\mathcal{F}_2$ are over $\textsc{Maj}^{-1}(0)$, $\textsc{Maj}^{-1}(1)$, $\textsc{Maj}^{-1}(1)$ respectively), define $\mathcal{D}_\zeta$ as the mixture over $y \sim \mathcal{F}_\zeta$ of $\mathcal{G}_y \coloneqq \mathcal{G}_{y_1} \times \cdots \times \mathcal{G}_{y_n}$ (i.e., $(x^1, \ldots, x^n) \sim \mathcal{G}_y$ is sampled by independently sampling $x^i \sim \mathcal{G}_{y_i}$ for all i). Put succinctly, $\mathcal{D}_\zeta \coloneqq \mathbb{E}_{y \sim \mathcal{F}_\zeta}[\mathcal{G}_y]$.

Now consider any conjunction C of width $w \le \frac{1}{16} n \log n$, and write $C(x^1, \ldots, x^n) = \prod_i C_i(x^i)$ where C_i is a conjunction. By Fact 9.*(i)*, $[n]$ can be partitioned into $A \cup B$ such that $C_i(\mathcal{G}_0) = 1$ for all $i \in A$, and $C_i(\mathcal{G}_0) = 0$ for all $i \in B$. Abbreviate $C_i(\mathcal{G}_1)$ as c_i, and for $S \subseteq [n]$ write $c_S \coloneqq \prod_{i \in S} c_i$. Identify $y \in \{0,1\}^n$ with $Y \coloneqq \{i : y_i = 1\}$, so $|y| = |Y|$. Let the uniform distribution over all size-s subsets of S be denoted by $\binom{S}{s}$, so $y \sim \mathcal{F}_\zeta$ corresponds to $Y \sim \binom{[n]}{n/2-1+\zeta}$. Let $I_{Y \supseteq B} \coloneqq \prod_{i \notin Y} C_i(\mathcal{G}_0)$ be the indicator random variable for the event $Y \supseteq B$. Now for $\zeta \in \{0,1,2\}$,

$$C(\mathcal{D}_\zeta) \;=\; \mathbb{E}_{y \sim \mathcal{F}_\zeta}[C(\mathcal{G}_y)] \;=\; \mathbb{E}_{y \sim \mathcal{F}_\zeta}\big[\textstyle\prod_i C_i(\mathcal{G}_{y_i})\big] \;=\; \mathbb{E}_{Y \sim \binom{[n]}{n/2-1+\zeta}}\big[c_Y \cdot I_{Y \supseteq B}\big]$$

$$= \underbrace{\mathbb{P}_{Y \sim \binom{[n]}{n/2-1+\zeta}}[Y \supseteq B]}_{p_\zeta} \cdot c_B \cdot \underbrace{\mathbb{E}_{S \sim \binom{A}{n/2-1+\zeta-|B|}}[c_S]}_{q_\zeta}.$$

If $c_B = 0$ then $C(\mathcal{D}_1) = 0$, so assume $c_B > 0$. Factoring out c_B and defining p_ζ and q_ζ as above (but q_ζ is undefined if $p_\zeta = 0$), our goal is to show that either $p_0 q_0 > \frac{1}{3} p_1 q_1$ or $p_2 q_2 > \frac{4}{3} p_1 q_1$ or $p_1 q_1 = 0$. There are three cases depending on whether $|B|$ is greater than, equal to, or less than $n/2$. First we collect some generally useful properties:

▷ Claim 10.

(i) $p_0 = \frac{n/2-|B|}{n/2} \cdot p_1$ and $p_1 = \frac{n/2+1-|B|}{n/2+1} \cdot p_2$.

(ii) $0 < q_1 \le \sqrt{n} \cdot q_2$ if q_1 is defined.

Proof. *(i)*: We just consider p_0 vs. p_1 since p_1 vs. p_2 is similar. Imagine sampling $Y_1 \sim \binom{[n]}{n/2}$ and then obtaining the set Y_0 by removing a uniformly random $i \in Y_1$. If $Y_1 \supseteq B$, then $Y_0 \supseteq B$ when $i \in Y_1 \setminus B$, which happens with probability $\frac{n/2-|B|}{n/2}$ (assuming $|B| \le n/2$; if $|B| > n/2$ then $p_0 = p_1 = 0$). Thus

$$p_0 \;=\; \mathbb{P}[Y_0 \supseteq B] \;=\; \mathbb{P}[Y_0 \supseteq B \,|\, Y_1 \supseteq B] \cdot \mathbb{P}[Y_1 \supseteq B] \;=\; \tfrac{n/2-|B|}{n/2} \cdot p_1.$$

(ii): Let w_i be the width of C_i, so $\sum_i w_i = w \le \frac{1}{16} n \log n$. Then $w_i \le \frac{1}{4} \log n \le m/4$ for at least $3n/4$ many values of i, and for such i note that by Fact 9.*(ii)*, $c_i \ge 3^{-(\log n)/4} \ge n^{-2/5}$ if $i \in A$. This implies that if we sample a uniformly random i from any $A' \subseteq A$ with $|A'| = n/2$ (note that $|A| \ge n/2$ if q_1 is defined) then $\mathbb{E}_{i \in A'}[c_i] \ge \frac{1}{2} \cdot n^{-2/5} + \frac{1}{2} \cdot 0 \ge 1/\sqrt{n}$. Now to relate q_2 and q_1,

$$q_2 \;=\; \mathbb{E}_{S \sim \binom{A}{n/2-|B|}}\big[c_S \cdot \mathbb{E}_{i \in A \setminus S}[c_i]\big] \;\ge\; \mathbb{E}_{S \sim \binom{A}{n/2-|B|}}\big[c_S/\sqrt{n}\big] \;=\; q_1/\sqrt{n}$$

where the inequality uses $|A \setminus S| = (n - |B|) - (n/2 - |B|) = n/2$. Furthermore, $q_1 > 0$ if q_1 is defined, because $n/2 - |B| \le |A| - n/4$ and thus there exists an $S \subseteq A$ with $|S| = n/2 - |B|$ and $c_i \ge n^{-2/5} > 0$ for all $i \in S$, hence $c_S > 0$. (A similar argument shows $0 < q_0 \le \sqrt{n} \cdot q_1$ if q_0 is defined, but we will not need that.) ◁

Case $|B| > n/2$. In this case, $p_1 = 0$ so we are done.

Case $|B| = n/2$. By Claim 10, $p_2 = p_1 \cdot (n/2 + 1)$ and $q_2 \ge q_1/\sqrt{n} > 0$ and thus

$$p_2 q_2 \;\ge\; p_1 q_1 \cdot (n/2+1)/\sqrt{n} \;>\; \tfrac{4}{3} p_1 q_1.$$

Case $|B| < n/2$. We will show that $\frac{p_0}{p_1} \geq \frac{1}{2} \cdot \frac{p_1}{p_2}$ and $\frac{q_2}{q_1} \geq \frac{9}{10} \cdot \frac{q_1}{q_0}$, which yields the punchline:

If $p_0 q_0 \leq \frac{1}{3} p_1 q_1$ then $\frac{q_2}{q_1} \geq \frac{9}{10} \cdot \frac{q_1}{q_0} \geq \frac{9}{10} \cdot 3 \cdot \frac{p_0}{p_1} \geq \frac{9}{10} \cdot 3 \cdot \frac{1}{2} \cdot \frac{p_1}{p_2} > \frac{4}{3} \cdot \frac{p_1}{p_2}$ and thus $p_2 q_2 > \frac{4}{3} p_1 q_1$.

First, $\frac{p_0}{p_1} \geq \frac{1}{2} \cdot \frac{p_1}{p_2}$ follows from Claim 10.*(i)* using $|B| \leq n/2 - 1$:

$$\frac{p_0}{p_1} \;=\; \frac{n/2+1}{n/2} \cdot \frac{n/2-|B|}{n/2+1-|B|} \cdot \frac{p_1}{p_2} \;\geq\; 1 \cdot \frac{n/2-(n/2-1)}{n/2+1-(n/2-1)} \cdot \frac{p_1}{p_2} \;=\; \frac{1}{2} \cdot \frac{p_1}{p_2}.$$

It just remains to show $\frac{q_2}{q_1} \geq \frac{9}{10} \cdot \frac{q_1}{q_0}$. Henceforth let $s \coloneqq n/2 - 1 - |B| \geq 0$. The experiment $S \sim \binom{A}{s+2}$ in the definition of q_2 can alternatively be viewed as:

- Sample $S_0 \sim \binom{A}{s}$.
- Sample $i \in A \setminus S_0$ u.a.r. and let $S_1 \coloneqq S_0 \cup \{i\}$.
- Sample $j \in A \setminus S_1$ u.a.r. and let $S = S_2 \coloneqq S_1 \cup \{j\}$.

That is, i and j are sampled without replacement. We consider an "ideal" (easier to analyze) version of this experiment that samples i and j with replacement, in other words, the third step becomes:

- Sample $j \in A \setminus S_0$ u.a.r. and let $S_2^* \coloneqq S_1 \cup \{j\}$.

Now S_2^* is a *multiset*, which may have two copies of i, in which case the product $c_{S_2^*}$ has two factors of c_i. Just as $q_2 \coloneqq \mathbb{E}[c_{S_2}]$, we let $q_2^* \coloneqq \mathbb{E}[c_{S_2^*}]$, and we next show how to derive $\frac{q_2^*}{q_1} \geq \frac{q_1}{q_0}$ from the following claim:

▷ Claim 11. For all nonnegative numbers $\alpha_1, \ldots, \alpha_N$ and $\beta_1, \ldots, \beta_N$ such that $\alpha_k \beta_k > 0$ for some k,

$$\frac{\sum_k \alpha_k \beta_k^2}{\sum_k \alpha_k \beta_k} \;\geq\; \frac{\sum_k \alpha_k \beta_k}{\sum_k \alpha_k}.$$

Proof. By clearing denominators, this inequality is equivalent to

$$\big(\textstyle\sum_k \alpha_k\big)\big(\sum_k \alpha_k \beta_k^2\big) \;\geq\; \big(\sum_k \alpha_k \beta_k\big)^2$$

which can be rewritten as

$$\textstyle\sum_{k,\ell} \alpha_k \alpha_\ell \beta_\ell^2 \;\geq\; \sum_{k,\ell} \alpha_k \beta_k \alpha_\ell \beta_\ell.$$

Subtracting $\sum_k \alpha_k^2 \beta_k^2$ from both sides, this is equivalent to

$$\textstyle\sum_{k<\ell} \big(\alpha_k \alpha_\ell \beta_\ell^2 + \alpha_\ell \alpha_k \beta_k^2\big) \;\geq\; \sum_{k<\ell} 2 \alpha_k \beta_k \alpha_\ell \beta_\ell.$$

We show that this inequality holds for each summand separately. Factoring out $\alpha_k \alpha_\ell$, this reduces to showing $\beta_\ell^2 + \beta_k^2 \geq 2 \beta_k \beta_\ell$, which holds since

$$\beta_\ell^2 + \beta_k^2 - 2 \beta_k \beta_\ell \;=\; (\beta_\ell - \beta_k)^2 \;\geq\; 0. \qquad \triangleleft$$

In the statement of Claim 11, let the index k correspond to S_0, let $N \coloneqq \binom{|A|}{s}$, let $\alpha_k \coloneqq c_{S_0}/N$, and let $\beta_k \coloneqq \mathbb{E}_{i \in A \setminus S_0}[c_i]$. Then

$$q_0 = \textstyle\sum_k \alpha_k \quad \text{and} \quad q_1 = \sum_k \alpha_k \beta_k \quad \text{and} \quad q_2^* = \sum_k \alpha_k \beta_k^2$$

and $q_0 \geq q_1 > 0$ by Claim 10.*(ii)* (i.e., $\alpha_k \beta_k > 0$ for some k) so by Claim 11 we indeed have $\frac{q_2^*}{q_1} \geq \frac{q_1}{q_0}$. To conclude that $\frac{q_2}{q_1} \geq \frac{9}{10} \cdot \frac{q_1}{q_0}$, we just need to show $q_2 \geq \frac{9}{10} q_2^*$.

The third step of the S_2 experiment is just the third step of the S_2^* experiment conditioned on $j \neq i$, which happens with probability $1 - \frac{1}{|A|-s}$. With probability $\frac{1}{|A|-s}$, we get $j = i$ in

the S_2^* experiment. If we condition on the latter event, it yields another experiment, whose result we call S_2^{err}, which is a multiset definitely containing two copies of i. Correspondingly we define $q_2^{\mathrm{err}} \coloneqq \mathbb{E}[c_{S_2^{\mathrm{err}}}]$ (with two factors of c_i). Now we have

$$q_2^* \;=\; \mathbb{P}[j \neq i]\cdot\mathbb{E}\big[c_{S_2^*} \,\big|\, j \neq i\big] + \mathbb{P}[j = i]\cdot\mathbb{E}\big[c_{S_2^*} \,\big|\, j = i\big] \;=\; \big(1 - \tfrac{1}{|A|-s}\big)\cdot q_2 + \tfrac{1}{|A|-s}\cdot q_2^{\mathrm{err}} \;\leq\; q_2 + \tfrac{2}{n}\cdot q_2^{\mathrm{err}}$$

since $|A| - s = (n - |B|) - (n/2 - 1 - |B|) = n/2 + 1 \geq n/2$.

The S_2^{err} experiment can alternatively be viewed as:

- Sample $S_1 \sim \binom{A}{s+1}$.
- Sample $i \in S_1$ u.a.r. and let $S_2^{\mathrm{err}} \coloneqq S_1 \cup \{i\}$.

This implies that $q_2^{\mathrm{err}} \leq q_1$ because the extra factor of $c_i \leq 1$ cannot increase the expectation. By Claim 10.*(ii)* we get $q_2^{\mathrm{err}} \leq q_1 \leq \sqrt{n}\cdot q_2$. Combining, we have

$$q_2^* \;\leq\; q_2 + \tfrac{2}{n}\cdot\sqrt{n}\cdot q_2 \;=\; \big(1 + \tfrac{2}{\sqrt{n}}\big)q_2 \;\leq\; \tfrac{10}{9}q_2$$

and thus $q_2 \geq \frac{9}{10}q_2^*$ as desired. This concludes the proof of Theorem 4.

4 Open questions

▶ **Open Question 12.** *Is there a total function* $g\colon \{0,1\}^m \to \{0,1\}$ *such that* $\mathsf{BPP}(\textsc{Xor}\circ g^n) \geq \Omega(n\log n\cdot\mathsf{BPP}(g))$ *or* $\mathsf{BPP}(\textsc{Maj}\circ g^n) \geq \Omega(n\log n\cdot\mathsf{BPP}(g))$*?*

Since Fact 9 captures the only properties of $g = \textsc{GapOr}$ used in our proof of Theorem 4, this provides a possible roadmap for confirming Open Question 12: Just find a total function g satisfying properties similar to Fact 9, enabling our proof of Theorem 4 to go through. However, such a g would need to have certificate complexity $\omega(\mathsf{BPP}(g))$, and it remains a significant open problem to find any such total function g (the "pointer function" [19, 2] and "cheat sheet" [1] methods do not seem to work).

Another approach for confirming Open Question 12 would be to generalize the strong direct sum theorem from [7] to show that $\mathsf{BPP}(\textsc{Xor}\circ g^n) \geq \Omega(n\cdot\overline{\mathsf{BPP}}_{1/n}(g))$ or $\mathsf{BPP}(\textsc{Maj}\circ g^n) \geq \Omega(n\cdot\overline{\mathsf{BPP}}_{1/n}(g))$ holds for all g. This would answer Open Question 12 in the affirmative, since [7] designed a total function g satisfying $\overline{\mathsf{BPP}}_{1/n}(g) \geq \Omega(\mathsf{RP}(g)\cdot\log n)$ using the "pointer function" method. Compared to our approach from the previous paragraph, this approach involves less stringent requirements on g, which makes it easier to design g but harder to prove the composition lower bound.

▶ **Open Question 13.** *Is there a total function* $f\colon \{0,1\}^n \to \{0,1\}$ *such that* $\mathsf{BPP}^*(f) \geq \omega(\mathsf{BPP}^\dagger(f))$ *(or similarly,* $\mathsf{BPP}(f\circ\textsc{GapMaj}^n) \geq \omega(\mathsf{BPP}(f\circ\textsc{GapOr}^n))$*)?*

It is not difficult to find such a *partial* function f. Namely, take any function $f'\colon \{0,1\}^n \to \{0,1\}$ such that $\mathsf{BPP}^*(f') \geq \Omega(n\log n)$, such as $f' = \textsc{Xor}$ or $f' = \textsc{Maj}$. Then take $f = f'\circ\textsc{Which}^n$, which has input length $2n$ (recall from Subsection 1.1 that given $y \in \{0,1\}^2$ with the promise that y has Hamming weight 1, $\textsc{Which}(y)$ indicates the location of the unique 1 in y). A simple reduction shows $\mathsf{BPP}^*(f) \geq \mathsf{BPP}^*(f')$. However, $\mathsf{BPP}^\dagger(f) \leq O(n)$: For each block of 2 bits, we can repeatedly query both until one of them returns 1 (which takes $O(1)$ queries in expectation). After doing this for all n blocks (which takes $O(n)$ queries in expectation), we know for sure what the entire actual input is. By Markov's inequality, we can abort the execution after $O(n)$ queries while introducing only a small constant error probability. (Intuitively, composition with $\textsc{Which}$ preserves hardness for 2-sided noise but converts 1-sided noise to "0-sided noise", and no partial function needs $\omega(n)$ queries in the setting of 0-sided noise.)

In communication (rather than query) complexity, somewhat analogous questions have been studied in specific contexts [24, 8, 26]. The proof of Theorem 2 also works for communication complexity. It would be interesting to develop analogues of Theorem 3 and Theorem 4 for communication complexity.

References

1 Scott Aaronson, Shalev Ben-David, and Robin Kothari. Separations in query complexity using cheat sheets. In *Proceedings of the 48th Symposium on Theory of Computing (STOC)*, pages 863–876. ACM, 2016. doi:10.1145/2897518.2897644.

2 Andris Ambainis, Kaspars Balodis, Aleksandrs Belovs, Troy Lee, Miklos Santha, and Juris Smotrovs. Separations in query complexity based on pointer functions. *Journal of the ACM*, 64(5):32:1–32:24, 2017. doi:10.1145/3106234.

3 Anurag Anshu, Dmitry Gavinsky, Rahul Jain, Srijita Kundu, Troy Lee, Priyanka Mukhopadhyay, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity. In *Proceedings of the 37th Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, pages 10:1–10:13. Schloss Dagstuhl, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.10.

4 Andrew Bassilakis, Andrew Drucker, Mika Göös, Lunjia Hu, Weiyun Ma, and Li-Yang Tan. The power of many samples in query complexity. In *Proceedings of the 47th International Colloquium on Automata, Languages, and Programming (ICALP)*. Schloss Dagstuhl, 2020. To appear.

5 Shalev Ben-David and Eric Blais. A tight composition theorem for the randomized query complexity of partial functions. Technical Report 2002.10809, arXiv, 2020. arXiv:2002.10809.

6 Shalev Ben-David and Robin Kothari. Randomized query complexity of sabotaged and composed functions. *Theory of Computing*, 14(1):1–27, 2018. doi:10.4086/toc.2018.v014a005.

7 Eric Blais and Joshua Brody. Optimal separation and strong direct sum for randomized query complexity. In *Proceedings of the 34th Computational Complexity Conference (CCC)*, pages 29:1–29:17. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.CCC.2019.29.

8 Eric Blais, Joshua Brody, and Badih Ghazi. The information complexity of hamming distance. In *Proceedings of the 18th International Workshop on Randomization and Computation (RANDOM)*, pages 465–489. Schloss Dagstuhl, 2014. doi:10.4230/LIPIcs.APPROX-RANDOM.2014.465.

9 Harry Buhrman and Ronald de Wolf. Complexity measures and decision tree complexity: A survey. *Theoretical Computer Science*, 288(1):21–43, 2002. doi:10.1016/S0304-3975(01)00144-X.

10 Chris Cade. Post-selected classical query complexity. Technical report, arXiv, 2018. arXiv:1804.10010.

11 Chinmoy Dutta and Jaikumar Radhakrishnan. Lower bounds for noisy wireless networks using sampling algorithms. In *Proceedings of the 49th Symposium on Foundations of Computer Science (FOCS)*, pages 394–402. IEEE, 2008. doi:10.1109/FOCS.2008.72.

12 William Evans and Nicholas Pippenger. Average-case lower bounds for noisy Boolean decision trees. *SIAM*, 28(2):433–446, 1998. doi:10.1137/S0097539796310102.

13 Uriel Feige, Prabhakar Raghavan, David Peleg, and Eli Upfal. Computing with noisy information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994. doi:10.1137/S0097539791195877.

14 Dmitry Gavinsky, Troy Lee, Miklos Santha, and Swagato Sanyal. A composition theorem for randomized query complexity via max-conflict complexity. In *Proceedings of the 46th International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 64:1–64:13. Schloss Dagstuhl, 2019. doi:10.4230/LIPIcs.ICALP.2019.64.

15 Dmitry Gavinsky and Shachar Lovett. En route to the log-rank conjecture: New reductions and equivalent formulations. In *Proceedings of the 41st International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 514–524. Springer, 2014. doi:10.1007/978-3-662-43948-7_43.

16 Mika Göös and T. S. Jayram. A composition theorem for conical juntas. In *Proceedings of the 31st Computational Complexity Conference (CCC)*, pages 5:1–5:16. Schloss Dagstuhl, 2016. doi:10.4230/LIPIcs.CCC.2016.5.

17 Mika Göös, T. S. Jayram, Toniann Pitassi, and Thomas Watson. Randomized communication vs. partition number. *ACM Transactions on Computation Theory*, 10(1):4:1–4:20, 2018. doi:10.1145/3170711.

18 Mika Göös, Shachar Lovett, Raghu Meka, Thomas Watson, and David Zuckerman. Rectangles are nonnegative juntas. *SIAM Journal on Computing*, 45(5):1835–1869, 2016. doi:10.1137/15M103145X.

19 Mika Göös, Toniann Pitassi, and Thomas Watson. Deterministic communication vs. partition number. *SIAM Journal on Computing*, 47(6):2435–2450, 2018. doi:10.1137/16M1059369.

20 Navin Goyal and Michael Saks. Rounds vs. queries tradeoff in noisy computation. *Theory of Computing*, 6(1):113–134, 2010. doi:10.4086/toc.2010.v006a006.

21 Rahul Jain, Hartmut Klauck, and Miklos Santha. Optimal direct sum results for deterministic and randomized decision tree complexity. *Information Processing Letters*, 110(20):893–897, 2010. doi:10.1016/j.ipl.2010.07.020.

22 Jedrzej Kaniewski, Troy Lee, and Ronald de Wolf. Query complexity in expectation. In *Proceedings of the 42nd International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 761–772. Springer, 2015. doi:10.1007/978-3-662-47672-7_62.

23 Claire Kenyon and Valerie King. On Boolean decision trees with faulty nodes. *Random Structures and Algorithms*, 5(3):453–464, 1994. doi:10.1002/rsa.3240050306.

24 Marco Molinaro, David Woodruff, and Grigory Yaroslavtsev. Beating the direct sum theorem in communication complexity with implications for sketching. In *Proceedings of the 24th Symposium on Discrete Algorithms*, pages 1738–1756. ACM-SIAM, 2013. doi:10.1137/1.9781611973105.125.

25 Ilan Newman. Computing in fault tolerant broadcast networks and noisy decision trees. *Random Structures and Algorithms*, 34(4):478–501, 2009. doi:10.1002/rsa.20240.

26 Mert Saglam. Near log-convexity of measured heat in (discrete) time and consequences. In *Proceedings of the 59th Symposium on Foundations of Computer Science (FOCS)*, pages 967–978. IEEE, 2018. doi:10.1109/FOCS.2018.00095.

27 Alexander Sherstov. Making polynomials robust to noise. *Theory of Computing*, 9:593–615, 2013. doi:10.4086/toc.2013.v009a018.

A Proof of Theorem 2: Or never necessitates amplification

For completeness, we provide a self-contained proof that $\mathsf{BPP}^*(\textsc{Or}) \leq O(n)$, using the following standard fact about random walks ("the drunkard at the cliff").

▶ **Lemma 14.** *Consider a random walk on the integers that begins at* 0 *and in each step moves right* $(+1)$ *with probability* p *and moves left* (-1) *with probability* $1-p$.

(i) *If* $p < 1/2$ *then the expected time at which the walk first visits* -1 *is* $1/(1-2p)$.

(ii) *If* $p > 1/2$ *then the probability that the walk ever visits* -1 *is* $(1-p)/p$.

Proof of Lemma 14. *(i)*: If random variable X represents the time at which the walk first visits -1, then its expectation satisfies $\mathbb{E}[X] = 1 + p \cdot 2\mathbb{E}[X]$ since after the first step, it either is already at -1, or is at $+1$ in which case to reach -1 it must first get back to 0 ($\mathbb{E}[X]$ expected time) then from there get to -1 (another $\mathbb{E}[X]$ expected time). This equation has a unique solution $\mathbb{E}[X] = 1/(1-2p) < \infty$.

(ii): If event E represents the walk ever visiting -1, then its probability satisfies $\mathbb{P}[E] = (1-p) \cdot 1 + p \cdot \mathbb{P}[E]^2$ since after the first step, it either is already at -1, or is at $+1$ in which case to reach -1 it must first get back to 0 (probability $\mathbb{P}[E]$) then from there get to -1 (again probability $\mathbb{P}[E]$). This equation has two solutions $\mathbb{P}[E] \in \{(1-p)/p, 1\}$. To rule out

$\mathbb{P}[E] = 1$, we define q_k as the probability that the walk visits -1 within the first k steps, and we show by induction on k that $q_k \leq (1-p)/p$. The base case is trivial since $q_0 = 0$. Assuming $q_k \leq (1-p)/p$ we show $q_{k+1} \leq (1-p)/p$. After the first step, with probability $1-p$ it is already at -1, and with probability p it is at $+1$. In the latter case, to get to -1 within a total of $k+1$ steps (including the first step), it must get from $+1$ to 0 and then from there it must get to -1, all within k more steps; in particular, the walk must get from $+1$ to 0 within k steps (probability $\leq q_k$) and then from 0 to -1 within k steps (probability $\leq q_k$). Overall we can bound $q_{k+1} \leq (1-p) \cdot 1 + p \cdot q_k^2 \leq (1-p) + p \cdot (1-p)^2/p^2 = (1-p)/p$. ◀

Proof of Theorem 2. We may assume the noise probabilities are $\leq 1/4$ (rather than just $\leq 1/3$), because whenever an input bit is queried, we can instead query it five times and pretend that the majority vote was the result of the single query. This would only affect the cost by a constant factor. With this assumption, here is our decision tree, on input $y \in \{0,1\}^n$:

For $i = 1, 2, \ldots, n$:
  Repeat:
    Query y_i.
    If the queries to y_i have resulted in more 0s than 1s so far,
      then break out of the inner loop.
    If a total of $6n$ queries have been made (across all input bits), then halt and output 1.
Halt and output 0.

This decision tree's cost is $\leq 6n$. To see the correctness, consider any input $y \in \{0,1\}^n$ and any tuple of noise probabilities $(\nu_1, \ldots, \nu_n)$ where each $\nu_i \leq 1/4$. For each i, the random variable

"number of 1s minus number of 0s, among the queries to y_i so far"

is a random walk with move-right probability $p_i = \nu_i \leq 1/4$ if $y_i = 0$ and $p_i = 1 - \nu_i \geq 3/4$ if $y_i = 1$, and which stops when it visits -1.

First assume $\textsc{Or}(y) = 0$. Then for each i, $y_i = 0$ and so by Lemma 14.*(i)*, the expected number of queries until the inner loop is broken is $1/(1-2p_i) \leq 2$. By linearity, the expected total number of queries until all n inner loops have been broken is $\leq 2n$, so by Markov's inequality this number of queries is $< 6n$ with probability $\geq 2/3$. Thus the decision tree outputs 0 with probability $\geq 2/3$.

Now assume $\textsc{Or}(y) = 1$. Then for some i, $y_i = 1$ and so by Lemma 14.*(ii)*, with probability $1 - (1-p_i)/p_i = 2 - 1/p_i \geq 2/3$ there would never be more 0s than 1s from the queries to y_i. In that case, the decision tree would never break out of the i^{th} inner loop, even if it were allowed to run forever. Thus the decision tree outputs 1 with probability $\geq 2/3$. ◀

On Multilinear Forms: Bias, Correlation, and Tensor Rank

Abhishek Bhrushundi[1]
Rutgers University, Piscataway, NJ, USA
abhishek.bhr@gmail.com

Prahladh Harsha[2]
Tata Institute of Fundamental Research, Mumbai, India
prahladh@tifr.res.in

Pooya Hatami[3]
Dept. of Computer Science & Engineering, The Ohio State University, Columbus, OH, USA
pooyahat@gmail.com

Swastik Kopparty
Dept. of Computer Science & Dept. of Mathematics, Rutgers University, Piscataway, NJ, USA
swastik.kopparty@gmail.com

Mrinal Kumar[4]
Dept. of Computer Science & Engineering, IIT Bombay, India
mrinalkumar08@gmail.com

Abstract

In this work, we prove new relations between the bias of multilinear forms, the correlation between multilinear forms and lower degree polynomials, and the rank[5] of tensors over $\mathbb{F}_2$. We show the following results for multilinear forms and tensors.

Correlation bounds. We show that a random d-linear form has exponentially low correlation with low-degree polynomials. More precisely, for $d = 2^{o(k)}$, we show that a random d-linear form $f(X_1, X_2, \ldots, X_d) : \left(\mathbb{F}_2^k\right)^d \to \mathbb{F}_2$ has correlation $2^{-k(1-o(1))}$ with any polynomial of degree at most $d/2$ with high probability.
This result is proved by giving near-optimal bounds on the bias of a random d-linear form, which is in turn proved by giving near-optimal bounds on the probability that a sum of t random d-dimensional rank-1 tensors is identically zero.

Tensor rank vs Bias. We show that if a 3-dimensional tensor has small rank then its bias, when viewed as a 3-linear form, is large. More precisely, given any 3-dimensional tensor

$$T : [k]^3 \to \mathbb{F}_2$$

of rank at most t, the bias of the 3-linear form

$$f_T(X_1, X_2, X_3) := \sum_{(i_1,i_2,i_3)\in[k]^3} T(i_1, i_2, i_3) \cdot X_{1,i_1} \cdot X_{2,i_2} \cdot X_{3,i_3}$$

is at least $(3/4)^t$.

[1] This work was done when the author was a graduate student at Rutgers University, USA
[2] This work was done when the author was visiting Rutgers University/DIMACS, USA and Weizmann Institute of Science, Israel.
[3] Part of this work was done when the author was a postdoc at DIMACS.
[4] Part of this work was done when the author was a postdoc at Center for Mathematical Sciences and Applications, Harvard University, USA.
[5] Here, "rank" refers to the standard notion of the rank of a tensor (***not*** analytic, slice, or partition park).


© Abhishek Bhrushundi, Prahladh Harsha, Pooya Hatami, Swastik Kopparty, and Mrinal Kumar; licensed under Creative Commons License CC-BY
Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 29; pp. 29:1–29:23

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

This bias vs tensor-rank connection suggests a natural approach to proving nontrivial tensor-rank lower bounds. In particular, we use this approach to give a new proof that the finite field multiplication tensor has tensor rank at least $3.52k$, which is the best known rank lower bound for any explicit tensor in three dimensions over $\mathbb{F}_2$. Moreover, this relation between bias and tensor rank holds for d-dimensional tensors for any fixed d.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures

Keywords and phrases polynomials, Boolean functions, tensor rank, bias, correlation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.29

Category RANDOM

Related Version `https://arxiv.org/abs/1804.09124`

Funding *Prahladh Harsha*: Supported by the the Department of Atomic Energy, Government of India, under project# RTI4001, the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467, and the Israel-India ISF-UGC grant.
Pooya Hatami: Supported by NSF grant CCF-1947546.
Swastik Kopparty: Supported in part by NSF grants CCF-1253886 and CCF-1540634.

Acknowledgements We would like to thank Suryateja Gavva for helpful discussions. We would like to thank Shubhangi Saraf for suggesting the idea for the proof of Lemma 10.

1 Introduction

This work is motivated by two fundamental questions regarding "explicit constructions" in complexity theory: finding functions uncorrelated with low degree polynomials, and finding tensors with high tensor rank.

Functions uncorrelated with low degree polynomials

The first question is that of finding an explicit function uncorrelated with low degree polynomials. More concretely, we seek functions $f : \mathbb{F}_2^n \to \mathbb{F}_2$ such that for every polynomial $P(X_1, \ldots, X_n) \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree at most ℓ (assume $\ell \approx n^{0.1}$ say),

$$\Pr_{x \in \mathbb{F}_2^n}[f(x) = P(x)] = \frac{1}{2} + \varepsilon_n \,,$$

where ε_n is exponentially small in n. It is well known that a random function f has this property with ε_n superpolynomially small (and even exponentially small); the challenge is to find an explicit function f.

A solution to this problem will have immediate applications in Boolean circuit complexity. It will give hard-on-average problems for $\mathrm{AC}^0(\oplus)$, and via the Nisan-Wigderson hardness vs. randomness technique [15], it will give pseudorandom generators against $\mathrm{AC}^0(\oplus)$ (improving upon analogous results for AC^0 from the late 1980s). The original motivation for an explicit function with small ε_n came from the seminal work of Razborov [17] and Smolensky [20] who used such functions to prove lower bounds against sub-exponential sized $\mathrm{AC}^0(\oplus)$ circuits. In particular, they showed that for the MOD_3 function $\varepsilon_n \leq \frac{1}{3} + O(1/\sqrt{n})$ and for the MAJORITY function $\varepsilon_n = O(\ell/\sqrt{n})$.[6]

[6] In a recent work [23], Viola showed that there exist degree ℓ polynomials which have correlation $\Omega(\ell/\sqrt{n})$ with the MAJORITY function, thereby showing that the aforementioned upper bound in [17, 20] are essentially tight.

The current best known constructions of explicit functions [17, 20, 3, 24] that cannot be approximated by low-degree polynomials come in two flavors, (a) polynomially small ε_n (in fact, $O(1/\sqrt{n})$) for large degree bounds (ℓ as large as $n^{0.1}$) or (b) exponentially small ε_n for small degree bounds ($\ell = o(\log n)$). However, we do not know of any explicit function f that exhibits exponentially small ε_n against low-degree polynomials of polynomially large (or even super-logarithmically large) degree polynomials. For a nice survey on correlation with low degree polynomials, see [22].

Tensors with high rank

The second question is that of finding an explicit tensor of high tensor rank. Tensors are a high-dimensional generalization of (2-dimensional) matrices. Just as a matrix of size k over a field $\mathbb{F}$ is given by a map $M : [k]^2 \to \mathbb{F}$, a tensor T of dimension d and size k is given by a map $T : [k]^d \to \mathbb{F}$. A tensor T is said to be of rank one if there exist vectors $u_1, u_2, \ldots, u_d \in \mathbb{F}_2^k$ such that $T = u_1 \otimes u_2 \otimes \cdots \otimes u_d$ or equivalently, for all $(i_1, \ldots, i_d) \in [k]^d$, we have $T(i_1, \ldots, i_d) = u_{1,i_1} \cdot u_{2,i_2} \cdots u_{d,i_d}$. A tensor T is said to be of tensor-rank at most t if it can be written as the sum of t rank one tensors. We seek tensors with tensor-rank as high as possible.

It is well known (and easy to prove) that a random tensor T has tensor rank t as large as $\Omega(k^{d-1}/d)$. The challenge is to find an explicit such T with tensor rank larger than $k^{\lfloor \frac{d}{2} \rfloor}$. A substantial improvement on this lower bound for any explicit tensor will have immediate applications in arithmetic circuit complexity; for $d = 3$, it will give improved arithmetic circuit lower bounds [21], and for large d it will give superpolynomial arithmetic formula lower bounds [16, 6]. For general *odd* d, a lower bound of $2k^{\lfloor d/2 \rfloor} + k - O(d \log k)$ was shown for an explicit tensor by Alexeev et al. [1], while for *even* d, no lower bounds better than the trivial bound $k^{\lfloor \frac{d}{2} \rfloor}$ are known for any explicit tensor.

Unlike matrix rank, we do not have a good understanding of tensor-rank even for 3-dimensional tensors. For instance, it is known that for a given 3-dimensional tensor T over the rationals, the problem of deciding if the rank of T is at most k is NP-hard [10]. In the case of dimension three, the tensor-rank of very specific tensors like the matrix multiplication tensor [4, 19], the finite field multiplication tensor [7, 18] and the polynomial multiplication tensor [5, 11] has been studied in prior works. For this case, the current best lower bound known for any explicit tensor over $\mathbb{F}_2$ is a lower bound of $3.52k$ for the finite field multiplication tensor due to Chudnovsky and Chudnovsky [7, 18], which builds on the lower bound result of Brown and Dobkin [5] for the polynomial multiplication tensor. For general fields, the best known lower bound for any explicit tensor is $2.5k - o(k)$ for the matrix multiplication tensor due to Bläser [4].

Also relevant to this discussion is a recent result of Effremenko et al. [8], who showed that a fairly general class of lower bound techniques called *rank methods* are not strong enough to give lower bounds on tensor rank stronger than $2^d \cdot k^{\lfloor d/2 \rfloor}$. In a nutshell, not only can we not prove good tensor rank lower bounds, we do not even have techniques, which "in principle" could be useful for such lower bounds!

1.1 Our results

We make contributions to both the above questions by studying *multilinear forms* and their *bias*. A d-linear form is a map $f : (\mathbb{F}_2^k)^d \to \mathbb{F}_2$ which is linear in each of its d arguments. The *bias* of a d-linear form is defined as follows.

$$\mathrm{bias}(f) := \left| \mathbb{E}_{x_1, \ldots, x_d \in \mathbb{F}_2^k} [(-1)^{f(x_1, \ldots, x_d)}] \right| .$$

This measures the difference between the probability of output 1 and output 0. Similarly, the correlation of a d-linear form f with another function g is defined as $\mathrm{corr}(f,g) := \mathrm{bias}(f-g)$, which measures the difference between the probabilities (on a random input) that f and g agree and disagree.

A d-linear form f can naturally be viewed as a polynomial of degree d in $n = kd$ variables. We can then ask, for some $\ell \ll d$, is there a d-linear form f such that the correlation of f with every degree ℓ polynomial in $\mathbb{F}_2[X_1, \ldots, X_n]$ is small? Knowing the existence of a d-linear f that achieves this small correlation property gives a significantly reduced search space for finding an explicit f with small correlation with lower degree polynomials. Our first result gives a positive answer to this question for a large range of ℓ and d.

▶ **Theorem** (A). *Let $d = o(n/\log n)$ and let $k = \frac{n}{d}$. Let $\ell < d/2$. Then with high probability, for a uniformly random d-linear form $f : (\mathbb{F}_2^k)^d \to \mathbb{F}_2$, we have that for all polynomial $P(X_1, \ldots, X_n) \in \mathbb{F}_2[X_1, \ldots, X_n]$ of degree at most ℓ:*

$$\mathrm{corr}(f, P) \leq 2^{-k(1-o(1))} = 2^{-\frac{n}{d}(1-o(1))}.$$

Moreover, for every d-linear form f, there is a degree 0 polynomial P (namely the constant 0 polynomial) such that $\mathrm{corr}(f, P) \geq \Omega(2^{-k})$.

For d small enough ($\tilde{O}(\log n)$), the above theorem actually holds with $\ell = d - 1$.

An important step towards proving Theorem A is a precise understanding of the distribution of the *bias* of a random d-linear form. Along the way, we give tight upper bounds on the probability that the sum of t random *rank-1* d-dimensional tensors equals 0.

Previously, a beautiful result of Ben-Eliezer, Lovett and Hod [2] showed that for all $d < \alpha n$, there are polynomials $f(X_1, \ldots, X_n)$ of degree d whose correlation with polynomials of degree $\ell = d-1$ is $2^{-\Omega(n/d)}$. The results are incomparable; the f in [2] need not come from a d-linear form, and for this more general setting the bound $2^{-\Omega(n/d)}$ might not be tight, but on the positive side [2] can handle larger d while proving correlation bounds against polynomials with degree as large as $d - 1$.

A d-linear form f can also be naturally represented as a d-dimensional tensor. Indeed, f can be completely specified by the tensor T of values $f(e_{i_1}, e_{i_2}, \ldots, e_{i_d})$, as the i_j vary in $[k]$. We can then ask, are there natural properties of the d-linear form f which would imply that the tensor rank of T is high? In our next main result, we prove a lower bound on the rank of a three dimensional tensor by studying the bias of the corresponding trilinear form. As far as we know, this is the first *analytic* property of low rank tensors which appears to be useful for lower bounds on tensor rank. Prior to this work, all the tensor rank lower bound proofs appear to be *algebraic*.

▶ **Theorem** (B). [7] *Let $f : (\mathbb{F}_2^k)^3 \to \mathbb{F}_2$ be a 3-linear form. Let T be the natural representation of f as a tensor (see above), and let t be the rank[8] of T. Then*

$$\mathrm{bias}(f) \geq \left(\frac{3}{4}\right)^t.$$

In particular, if $\mathrm{bias}(f) = 2^{-(1-o(1))k}$, then $t \geq k \cdot \log_{\frac{4}{3}} 2$. Moreover, for every t there is a tensor T with tensor rank t such that the following is true.

$$\mathrm{bias}(f) \leq \left(\frac{3}{4}\right)^t + \frac{3}{2^k}.$$

[7] For brevity, we state this theorem here for $d = 3$, but it holds more generally for higher dimensional tensors as well. See Section 3 for details.

[8] Here, "rank" refers to the standard notion of the rank of a tensor.

This gives a natural and clean route to proving nontrivial tensor rank lower bounds for explicit 3 dimensional tensors. In particular, trilinear forms with nearly minimal bias of of $2^{-(1-o(1))k}$ must have tensor rank at least $2.409k$ (which happens to be tight). A finer analysis of our arguments shows that trilinear forms with *exactly* minimal bias of $\approx 2 \cdot 2^{-k}$, such as the finite field multiplication tensor, have tensor rank $\geq 3.52k$, thus matching the best known explicit tensor rank lower bound for 3-dimensional tensors [5, 7, 18] for *any* explicit tensor. It also immediately implies that the matrix multiplication tensor has tensor rank $\geq 1.8k$, which is nontrivial (but still far from the best known bound of $3k$ [19, 4]). We remark that since every a 3-linear form $f : (\mathbb{F}_2^k)^d \to \mathbb{F}_2$ has bias at least $\exp(-\Omega(k))$, we cannot hope to prove super linear lower bounds on tensor rank via a direct use of this connection between bias and tensor rank. We also note while that an analogous connection between bias and tensor rank also holds in 4 and higher dimensions (see Theorem 8), the quantitative bounds are not strong enough to give a non-trivial lower bound on tensor rank for d dimensional tensors for $d \geq 4$. Here, by *non-trivial* tensor rank lower bounds, we mean bounds are better than the bound of $k^{\lfloor d/2 \rfloor}$ that can be obtained by just flattening the tensor into a matrix and using the rank of the matrix as a lower bound on tensor rank.

We remark this method of studying the bias of a three dimensional tensor as a tool for proving tensor rank lower bounds appears to be new. Informally it shows a non-trivial connection between one of the weakest measures of computational pseudorandomness of a function, namely bias, and one of the strongest measures of computational pseudorandomness, namely tensor rank. While such a connection is well known for matrices, to the best of our knowledge, this connection between bias and rank is new for tensors of dimension 3 and larger. In addition to the intrinsic appeal, this connection lets us recover the lower bound of $3.52k$ for an explicit three dimensional tensor over $\mathbb{F}_2$. To recover this lower bound, we end up using the proof of Theorem B in a non-blackbox manner.

The results of Theorem B can also be phrased in terms of the notion of *analytic rank* introduced in the work of Gowers and Wolf [9]. The analytic rank of a multilinear form f over a finite field $\mathbb{F}$ is defined by:

$$\operatorname{arank}(f) := -\log_{|\mathbb{F}|}(\operatorname{bias}(f)) \ .$$

Stated in this language[9], our result says that if $f : (\mathbb{F}_2^k)^3 \to \mathbb{F}_2$ is a 3-linear form of tensor rank t then

$$t \geq \frac{1}{\log_2(8/7)} \cdot \operatorname{arank}(f) \ .$$

This is essentially the best lower bound one can hope to prove on the tensor rank in terms of the analytic rank.
In their work, Gowers and Wolf prove that analytic rank is approximately subadditive. In particular, they show that

$$\operatorname{arank}(f+g) \leq 2^d \left(\operatorname{arank}(f) + \operatorname{arank}(g)\right) \ .$$

This implies only a quantitatively much weaker version of Theorem B which does not give any nontrivial tensor rank lower bounds even for $d = 3$. Shortly after we posted our paper online, Lovett [13] showed that analytic rank is *fully* subadditive (improving upon the above

[9] Similar to Theorem B, this also holds for general d dimensional tensors. We focus on the $d = 3$ case here.

result of Gowers and Wolf by getting rid of the multiplicative factor of 2^d). The proof is extremely elegant and clever. This result of Lovett implies and greatly elucidates the real reason underlying Theorem B, although we do not know if a tensor rank lower bound of anything close to $3.52k$ can be recovered directly from it.

1.2 Organization

Section 2 contains the preliminaries. Section 3 discusses the connection between bias and tensor rank (Theorem B above) and proves rank lower bounds for explicit tensors. Section 4 proves correlation bounds for random d-linear forms (Theorem A above) and other related results.

2 Preliminaries

Unless otherwise stated, we always work over the field $\mathbb{F}_2$. We use capital X, Y, Z etc. to denote formal variables or sets of formal variables, and small letters x, y, z to denote instantiations of these formal variables.

For integers $n, d \geq 0$, denote by $\mathrm{Poly}(n, d)$ the set of all degree $\leq d$ multilinear polynomials in $\mathbb{F}_2[X]$, where $X = \{X_1, ..., X_n\}$ is a variable set. Note that every $f \in \mathrm{Poly}(n, d)$ naturally corresponds to a unique map $f : \mathbb{F}_2^n \to \mathbb{F}_2$.

2.1 Bias and Correlation

Two fundamental notions used in this paper are those of bias and correlation, which we now define.

▶ **Definition 1** (Bias). *Bias of a function $f : \mathbb{F}_2^n \to \{0, 1\}$ is defined as*

$$\mathrm{bias}(f) := \left| \mathbb{E}_{x \in \mathbb{F}_2^n} (-1)^{f(x)} \right| .$$

The bias of an $\mathbb{F}_2$-valued function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is defined as $\mathrm{bias}(f) := \mathrm{bias}(\iota(f))$, where ι is the standard map from $\mathbb{F}_2$ to $\{0, 1\}$.

▶ **Definition 2** (Correlation). *We define the correlation between two functions $f, g : \mathbb{F}_2^n \to \mathbb{F}_2$, by*

$$\mathrm{corr}(f, g) := \mathrm{bias}(f - g) .$$

Given a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$, we will be interested in its maximum correlation with low degree polynomials. Towards this we define

$$\mathrm{corr}(f, d) := \max_{g \in \mathrm{Poly}(n,d)} \mathrm{corr}(f, g) .$$

More generally, given a class $\mathcal{C}$ of functions, we define

$$\mathrm{corr}(f, \mathcal{C}) := \max_{g \in \mathcal{C}} \mathrm{corr}(f, g) .$$

2.2 Tensors and d-linear forms

Tensors are generalizations of matrices to higher dimensions.

▶ **Definition 3** (Tensors and Tensor rank). *Let k and d be natural numbers. A d dimensional tensor T of size k over a field $\mathbb{F}$ is a map $T : [k]^d \to \mathbb{F}$. T is said to be of rank one if there exist d vectors $u_1, u_2, \ldots, u_d : [k] \to \mathbb{F}$ such that for every $(i_1, i_2, \ldots, i_d) \in [k]^d$, $T(i_1, i_2, \ldots, i_d) = \prod_{j=1}^{d} u_j(i_j)$. The rank of T is the minimum t such that T can be written as a sum of t rank one tensors.*

Every matrix can be naturally associated with a bilinear polynomial, and in some cases, one can study the properties of this bilinear polynomial as a proxy of studying various properties of the matrix itself. This paradigm also generalizes to tensors, as the following definition indicates.

▶ **Definition 4** (Tensors as Multilinear Forms). *Let $T : [k]^d \to \mathbb{F}$ be a d dimensional tensor. Then, the* set-multilinear *polynomial associated with T is the polynomial f_T in variables $\{X_{i,j} : i \in [d], j \in [k]\}$ over $\mathbb{F}$ defined as follows.*

$$f_T(X_{1,1}, X_{1,2}, \ldots, X_{d,k}) = \sum_{(i_1, i_2, \ldots, i_d) \in [k]^d} T(i_1, i_2, \ldots, i_d) \cdot \prod_{j=1}^{d} X_{j,i_j}.$$

Given the above association between d-dimensional tensors and d-linear forms, we will use the terms tensor and d-linear form interchangeably.

2.3 Some explicit tensors

We now define some explicit tensors which we shall use in the next section.

2.3.1 Trace tensor

▶ **Definition 5.** $\mathrm{Trace} : \mathbb{F}_{2^k} \to \mathbb{F}_2$ *is the $\mathbb{F}_2$-linear map defined as follows.*

$$\mathrm{Trace}(\alpha) = \alpha + \alpha^2 + \ldots + \alpha^{2^{k-1}}.$$

The Trace map will be useful for us as we define the candidate hard tensor for our lower bounds.

▶ **Definition 6.** *Let $Tr : \mathbb{F}_2^{k \times k \times k} \to \mathbb{F}_2$ be the function defined as follows.*

$$Tr(X, Y, Z) := \mathrm{Trace}(XYZ),$$

where XYZ denotes multiplication over the larger field $\mathbb{F}_{2^k}$ when $X = (X_1, X_2, \ldots, X_k), Y = (Y_1, Y_2, \ldots, Y_k), Z = (Z_1, Z_2, \ldots Z_k)$ are viewed as encodings of elements in $\mathbb{F}_{2^k}$.

Since Trace is an $\mathbb{F}_2$-linear map, the function $Tr(X, Y, Z)$ can be viewed as a 3-linear polynomial in the variables $X = (X_1, X_2, \ldots, X_k), Y = (Y_1, Y_2, \ldots, Y_k), Z = (Z_1, Z_2, \ldots Z_k)$. For the rest of this paper, when we say $Tr(X, Y, Z)$, we refer to this natural 3-linear polynomial and the three dimensional tensor associated with it. Up to a change of basis, this is the finite field multiplication tensor, which was analyzed by Chudnovsky-Chudnovsky [7] and Shparlinksi-Tsfasman-Vladut [18]. It is also worth noting that these papers also proved a surprising and beautiful $O(k)$ *upper bound* on the tensor rank of this tensor.

2.3.2 Matrix multiplication tensor

▶ **Definition 7.** *The tensor corresponding to the product of two $n \times n$ matrices is defined as*

$$M_n(X, Y, Z) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} X_{i,j} Y_{j,k} Z_{i,k} \,.$$

Here, $X = \{X_{i,j} : i, j \in [n]\}, Y = \{Y_{i,j} : i, j \in [n]\}, Z = \{Z_{i,j} : i, j \in [n]\}$.

Note that $M_n(X, Y, Z)$ is the trace of the matrix product $X \cdot Y \cdot Z^T$. In other words, $M_n(X, Y, Z^T) = \text{Trace}(X \cdot Y \cdot Z)$. Note this is the matrix trace and is different from the trace function considered in the previous section where we viewed X, Y, Z as elements of the large field.

3 High-rank tensors from unbiased polynomials

It is well-known that the bias of a bilinear form corresponding to a matrix $M \in \mathbb{F}_2^{k \times k}$ is tightly related to its rank $\text{rank}(M)$ (more precisely, $\text{bias}(M) = 2^{-\text{rank}(M)}$). In this section, we explore a similar connection for higher dimensional tensors. We then use this to (re)prove some existing tensor rank lower bounds (e.g., for the trace tensor and the matrix multiplication tensor). We note that while in the introduction we stated this connection between bias and tensor rank specifically for three dimensional tensors, we prove a general statement which holds even for higher dimensional tensors.

3.1 Small Bias implies large tensor rank

We begin with the main theorem of this section which shows tensors with small bias have large rank.

▶ **Theorem 8** (Small bias implies large rank)**.** *Let $P \in \mathbb{F}_2^{k \times k \cdots \times k}$ be any d-dimensional tensor of rank $\leq t$. Then*

$$\text{bias}(P) \geq \left(1 - \frac{2}{2^d}\right)^t \,.$$

An important ingredient of our proof will be the following lemma.

▶ **Lemma 9.** *Let d be a natural number. Let $M_1, M_2, \ldots, M_t \in \mathbb{F}_2^{k \times k \cdots \times k}$ be d-dimensional tensors of rank at most 1. Then,*

$$\Pr_{x_1, x_2, \ldots, x_d \in \mathbb{F}_2^k} [\forall i \in [t], M_i(x_1, x_2, \ldots, x_d) = 0] \geq \left(1 - \frac{1}{2^d}\right)^t \,. \tag{1}$$

Proof. Our proof is by induction on d.

Base Case. The base case when $d = 1$ trivially follows since if there are t linear forms $u_1, u_2, \ldots, u_t$ over $\mathbb{F}_2$, then the maximum number r of independent linear forms among them is at most t. We hence have,

$$\Pr_{x \in \mathbb{F}_2^k} [\forall i \in [t], u_i(x) = 0] = (1/2)^r \geq (1/2)^t \,. \tag{2}$$

Induction Step. For the inductive step, we assume that the lemma is true up to dimension $d-1$, and prove it for d dimensions. For every $i \in [t]$, we denote by u_i the linear form in $\mathbb{F}_2^k$ and by M_i' the $d-1$ dimensional tensor of rank 1 in $\mathbb{F}_2^{k \times k \times k \cdots \times k}$ such that

$$M_i(X_1, X_2, \ldots, X_d) = u_i(X_1) \cdot M_i'(X_2, X_3, \ldots, X_d)\,.$$

For every $S \subseteq [t]$, M_S denotes the tensor $\sum_{j \in S} M_j$, which has rank at most $|S|$. We now proceed via a sequence of inequalities.

$$\begin{aligned}
&\Pr_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} [\forall i \in [t], M_i(x_1, x_2, \ldots, x_d) = 0] \\
&= \mathbb{E}_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} \left[\prod_{i=1}^{t} \left(\frac{1 + (-1)^{M_i(x_1,x_2,\ldots,x_d)}}{2} \right) \right] \\
&= \mathbb{E}_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} \left[\frac{1}{2^t} \cdot \sum_{S \subseteq [t]} (-1)^{M_S(x_1,x_2,\ldots,x_d)} \right] \\
&= \mathbb{E}_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} \left[\mathbb{E}_{S \subseteq [t]} \left[(-1)^{M_S(x_1,x_2,\ldots,x_d)} \right] \right] \\
&= \mathbb{E}_{S \subseteq [t]} \left[\mathbb{E}_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} \left[(-1)^{M_S(x_1,x_2,\ldots,x_d)} \right] \right] .
\end{aligned}$$

Now, observe that for every $S \subseteq [t]$,

$$\mathbb{E}_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} \left[(-1)^{M_S(x_1,x_2,\ldots,x_d)} \right] \geq \Pr_{x_2,x_3,\ldots,x_d} \left[\forall j \in S, M_j'(x_2, x_3, \ldots, x_d) = 0 \right] .$$

Moreover, from the induction hypothesis, we get that for all $S \subseteq [t]$,

$$\Pr_{x_2,x_3,\ldots,x_d} \left[\forall j \in S, M_j'(x_2, x_3, \ldots, x_d) = 0 \right] \geq \left(1 - \frac{1}{2^{d-1}} \right)^{|S|} .$$

Plugging this back in the calculations, we get

$$\begin{aligned}
\Pr_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} [\forall i \in [t], M_i(x_1, x_2, \ldots, x_d) = 0] &\geq \mathbb{E}_{S \subseteq [t]} \left[\left(1 - \frac{1}{2^{d-1}} \right)^{|S|} \right] \\
&\geq \frac{1}{2^t} \cdot \left(1 + 1 - \frac{1}{2^{d-1}} \right)^t = \left(1 - \frac{1}{2^d} \right)^t . \quad \blacktriangleleft
\end{aligned}$$

We now complete the proof of Theorem 8.

Proof of Theorem 8. Since P has rank $\leq t$, then there is a collection of linear forms $u_1, u_2, \ldots, u_t$ and tensors $M_1, M_2, \ldots, M_t$ of rank at most 1 in $d-1$ dimensions such that

$$P(X_1, X_2, \ldots, X_d) = \sum_{i=1}^{t} u_i(X_1) \cdot M_i(X_2, X_3, \ldots, X_d)\,.$$

Now, observe that

$$\begin{aligned}
\mathrm{bias}(P) &= \left| \mathbb{E}_{x_1,x_2,\ldots,x_d \in \mathbb{F}_2^k} \left[(-1)^{P(x_1,x_2,\ldots,x_d)} \right] \right| \\
&= \Pr_{x_2,x_3,\ldots,x_d \in \mathbb{F}_2^k} [P(X_1, x_2, x_3, \ldots, x_d) \equiv 0] \\
&\geq \Pr_{x_2,x_3,\ldots,x_d \in \mathbb{F}_2^k} [\forall i \in [t],\ M_i(x_2, x_3, \ldots, x_d) = 0] \\
&\geq \left(1 - \frac{1}{2^{d-1}} \right)^t \qquad [\text{By Lemma 9}]\,. \quad \blacktriangleleft
\end{aligned}$$

We can complement the above theorem with an almost matching upper bound on the bias of random high rank tensors. It is known that a random high rank tensor has low bias. The following lemma gives a precise quantitative version of this observation (the idea for the proof was suggested to us by Shubhangi Saraf).

▶ **Lemma 10.** *For $i \in [t]$ and $j \in [d]$, let $u_{i,j} \in \mathbb{F}_2^k$ be a uniformly random vector. Consider the random rank-t d-linear form $p : (\mathbb{F}_2^k)^d \to \mathbb{F}_2$ given by*

$$p(x_1, x_2, \ldots, x_d) = \sum_{i=1}^{t} \prod_{j=1}^{d} \langle x_j, u_{i,j} \rangle \ .$$

Then

$$\mathbb{E}[\mathrm{bias}(p)] \leq d \cdot 2^{-k} + \left(1 - \frac{2}{2^d}\right)^t \ .$$

We refer the reader to the full version of our paper for the proof.

The following special cases of Theorem 8, for $d = 2$ and $d = 3$ will be useful for us, on our way to proving lower bounds on the rank of three dimensional tensors.

▶ **Corollary 11.** *Let $P \in \mathbb{F}_2^{k \times k}$ be a matrix of rank $\leq t \leq k$. Then,* $\mathrm{bias}(P) \geq 2^{-t}$.

▶ **Corollary 12.** *Let $P \in \mathbb{F}_2^{k \times k \times k}$ be a 3-dimensional tensor of rank $\leq t$. Then,* $\mathrm{bias}(P) \geq \left(\frac{3}{4}\right)^t$.

In the subsequent two sections, we will observe that some well-known explicit tensors in three dimensions have very low bias, and then use the above corollaries to conclude that these tensors have large rank.

3.2 A 3.52k Tensor Rank Lower Bound for $\mathrm{Trace}(XYZ)$

In this section, we use the bias-vs-tensor-rank connection explored in the previous section to construct explicit 3-dimensional tensors with large tensor rank.

It can be observed that $\mathrm{Trace}(XYZ)$ is a function with bias exactly $2/2^k - 1/2^{2k}$ (We omit the proof in interest of space).

▶ **Lemma 13.** $\mathrm{bias}(Tr(X, Y, Z)) = 2 \cdot 2^{-k} - 2^{-2k}$.

This lemma coupled with Corollary 12 immediately gives the following lower bound on tensor rank of $Tr(X, Y, Z)$.

▶ **Corollary 14.** $\mathrm{rank}(Tr(X, Y, Z)) \geq (\log_{4/3} 2) \cdot k \geq 2.409k$.

We remark that a much stronger rank lower-bound of $3.52k$ is known due to Chudnovsky and Chudnovsky [7, 18] and indeed we do a more careful analysis of our ideas to get a new proof of the $3.52k$ lower bound. We will need the following well-known rate-distance MRRW tradeoff for linear codes.

▶ **Theorem 15** ([14])**.** *Let S be a subspace of dimension at least k of $\mathbb{F}_2^t$, such that every non-zero vector in S has weight at least k. Then, $t \geq 3.52k$.*[10]

[10] The MRRW bound for binary codes states that any family of codes with fractional distance δ satisfies $R(\delta) \leq h_2\left(\frac{1}{2} - \sqrt{\delta(1-\delta)}\right)$ where $h_2(x) = x \log_2(1/x) + (1-x)\log_2(1/1-x)$ is the binary entropy function. The above mentioned bound can be obtained from this (see [5] for details).

▶ **Theorem 16.** *The rank of the tensor $Tr(X,Y,Z)$ is at least $3.52k$.*

Proof. Let the tensor rank of $Tr(X,Y,Z)$ be t. Then there exists t vectors $a_1, a_2, \ldots, a_t \in \mathbb{F}_2^k$ and t rank-1 matrices $M_1, M_2, \ldots, M_t$ such that

$$Tr(X,Y,Z) = \sum_{i=1}^{t} \langle a_i, X\rangle \cdot \langle Y, M_i Z\rangle \ . \tag{3}$$

Let A be the $k \times t$ matrix such that for every $i \in [t]$, the i^{th} column of A equals a_i. Let K be the kernel of A. Clearly, $\dim(K) \geq t-k$. In fact, $\dim(K) = t-k$. To see this, observe that if $\dim(K) \geq t-k+1$, then by the rank-nullity theorem, $\mathrm{rank}(A) \leq k-1$. Thus, there is a non-zero $x \in F_2^k$ denoted by x_0 such that for every $i \in [t]$, $\langle a_i, x_0\rangle = 0$. Thus, $Tr(x_0, Y, Z) \equiv 0$ for a non-zero x_0, which is a contradiction.

From the proof of Theorem 8 for $d = 3$, we know that

$$\mathrm{bias}(Tr(X,Y,Z)) = \Pr_{y,z\in\mathbb{F}_2^k}[Tr(X,y,z) = 0]\,.$$

So far we were proving a lower bound on $\Pr_{y,z\in\mathbb{F}_2^k}[Tr(X,y,z)=0]$ by proving a lower bound on $\Pr_{y,z\in\mathbb{F}_2^k}[\forall i \in [t], \langle y, M_i z\rangle = 0]$. Clearly, this seems to be somewhat lossy since even for a choice of y and z in $\mathbb{F}_2^k$ such that $\langle y, M_i z\rangle \neq 0$ for some $i \in [t]$, it is conceivable that $Tr(X,y,z)$ is identically zero. For this proof, we try to be a bit more careful about this. Note that for every $u \in K \subset \mathbb{F}_2^t$,

$$\sum_{i=1}^{t} u_i \cdot \langle a_i, X\rangle \equiv 0\,.$$

Thus, we have,

$$\begin{aligned}
\Pr_{y,z\in\mathbb{F}_2^k}[Tr(X,y,z)=0] &= \sum_{u\in K} \Pr_{y,z\in\mathbb{F}_2^k}[\forall i\in[t], \langle y, M_i z\rangle = u_i] \\
&= \sum_{u\in K} \mathbb{E}_{y,z}\left[\prod_{i\in[t]}\left(\frac{1+(-1)^{\langle y,M_i z\rangle + u_i}}{2}\right)\right] \\
&= \sum_{u\in K} \mathbb{E}_{y,z}\left[\mathbb{E}_{S\subseteq[t]}(-1)^{\langle y, M_S z\rangle}\cdot(-1)^{\langle u, 1_S\rangle}\right]\,.
\end{aligned}$$

Here, for every $S \subseteq [t]$, 1_S is the characteristic vector of S in t dimensions, and $M_S = \sum_{i\in S} M_i$. Simplifying further, we get,

$$\Pr_{y,z\in\mathbb{F}_2^k}[Tr(X,y,z)=0] = \mathbb{E}_{S\subseteq[t]}\left[\left(\mathbb{E}_{y,z}(-1)^{\langle y,M_S z\rangle}\right)\cdot\left(\sum_{u\in K}(-1)^{\langle u,1_S\rangle}\right)\right]\,.$$

Now, we observe that the term $\left(\sum_{u\in K}(-1)^{\langle u,1_S\rangle}\right) = |K|$ if and only if $1_S \in K^\perp$, otherwise it equals zero. Also, from Corollary 11, we know that $\left(\mathbb{E}_{y,z}(-1)^{\langle y,M_S z\rangle}\right) = 2^{-\mathrm{rank} M_S}$ is at at least $\max\{2^{-k}, 2^{-|S|}\}$. Plugging these into the inequality above, we have the following inequality (Below, $|v|$ denotes the Hamming weight of v).

$$\begin{aligned}
\Pr_{y,z\in\mathbb{F}_2^k}[Tr(X,y,z)=0] &\geq \frac{|K|}{2^t}\cdot\sum_{v\in K^\perp}\max\{2^{-k}, 2^{-|v|}\} \\
&\geq \mathbb{E}_{v\in K^\perp}\max\{2^{-k}, 2^{-|v|}\} \qquad [\text{ Since } |K|\cdot|K^\perp| = 2^t]
\end{aligned}$$

Recall that the dimension of $K^\perp$ equals k. Now,

$$\mathbb{E}_{v\in K^\perp} \max\{2^{-k}, 2^{-|v|}\} = 2^{-k} + \mathbb{E}_{v\in K^\perp\setminus\{0^k\}} \max\{2^{-k}, 2^{-|v|}\}\,.$$

From Lemma 13, we know that the bias of $Tr(X,Y,Z)$ is $2\cdot 2^{-k} - 2^{-2k}$. Thus, it must be the case that $\mathbb{E}_{v\in K^\perp\setminus\{0^k\}} \max\{2^{-k}, 2^{-|v|}\} \le (1-2^{-k})\cdot 2^{-k}$. But this is possible only if all the vectors in $K^\perp \setminus \{0^k\}$ have weight at least k. In this case, the space $K^\perp$ is a linear subspace of $\mathbb{F}_2^t$ of dimension k such that every non-zero vector in it has Hamming weight at least k. From Theorem 15, we get that $t \ge 3.52k$. This completes the proof. ◀

3.3 Lower Bound on the Rank of Matrix Multiplication Tensor

In this section, we obtain a lower bound on the rank of the matrix multiplication tensor by proving an upper bound on its bias. Even though better bounds are known for this tensor, our proof is a fairly straightforward application of our techniques, and we believe this is instructive.

Our main technical observation in this section is the following lemma which gives an upper bound on the bias of $M_n(\overline{X},\overline{Y},\overline{Z})$ as each of the variables take values in $\mathbb{F}_2$.

▶ **Lemma 17.** *The bias of $M_n(\overline{X},\overline{Y},\overline{Z})$ is at most $n\cdot 2^{-\frac{3n^2}{4}}$.*

Before proving Lemma 17, we note that Lemma 17 and Corollary 12 immediately imply a non-trivial lower bound on the tensor rank of M_n. .

▶ **Theorem 18.** *The tensor rank of M_n is at least $\frac{3n^2}{4\log_2(4/3)} \ge 1.8n^2$.*

Proof of Lemma 17. We observe that for any two fixed matrices x, y, the 3-linear form M_n reduces to a linear form in z which is non-zero iff the product of the two matrices x and y is non-zero. Furthermore, given a matrix y, the probability (over x) that the product matrix $x\cdot y$ is zero is exactly $2^{-n\cdot\mathrm{rank}(y)}$. Combining these observations, we have

$$\begin{aligned}\mathrm{bias}(M_n) &= \Pr_{x,y}[x\cdot y = 0_{n\times n}]\\ &= \mathbb{E}_y\left[2^{-n\cdot\mathrm{rank}(y)}\right]\\ &= \sum_{r=0}^{n}\Pr_y[\mathrm{rank}(y)=r]\cdot 2^{-nr}\,.\end{aligned}$$

To complete the proof, we rely on the following claim, whose proof we defer to the end of this section.

▷ Claim 19. For every $r\in\{0,1,\ldots,n\}$, the following inequality is true.

$$\Pr_y[\mathrm{rank}(y)=r] \le 2^{-(n-r)^2}\,.$$

From the claim above, we get

$$\begin{aligned}\mathrm{bias}(M_n) &\le \sum_{r=0}^{n} 2^{-(n-r)^2-nr} \\ &\le \sum_{r=0}^{n} 2^{-n^2-r^2+nr} \\ &\le 2^{-n^2}\sum_{r=0}^{n} 2^{r(n-r)} \\ &\le 2^{-n^2} n \cdot 2^{n^2/4} \\ &\le n \cdot 2^{-3n^2/4} .\end{aligned}$$

◀

For completeness, we now provide a proof of Claim 19. We remark that the following tighter bound is known (see [12, Theorem 3.2.1]).

$$\begin{aligned}\Pr_y[\mathrm{rank}(y) = r] &= 2^{-(n-r)^2} \cdot \prod_{i=n-r+1}^{n}\left(1-\frac{1}{2^i}\right) \cdot \left(\sum_{0\le i_1\le \ldots i_{n-r}\le r} \frac{1}{2^{i_1+\ldots+i_{n-r}}}\right) \\ &\le 2^{-(n-r)^2} \cdot \prod_{i=n-r+1}^{n}\left(1-\frac{1}{2^i}\right) \cdot \prod_{i=1}^{n-r}\left(1-\frac{1}{2^i}\right)^{-1} .\end{aligned}$$

However, the weaker bound given in the claim suffices for our purposes.

Proof of Claim 19. The goal is to upper bound the probability that a uniformly random $n \times n$ matrix y over $\mathbb{F}_2$ has rank equal to r. This probability is upper bounded by the probability that the rows of y are contained within a subspace of dimension r of $\mathbb{F}_2^n$. For any fixed subspace S of dimension equal to r, this event happens with a probability equal to $2^{-n(n-r)}$. The number of subspaces of $\mathbb{F}_2^n$ of dimension equal to r is given by the Gaussian binomial coefficient $\begin{bmatrix} n \\ r \end{bmatrix}_2 = \prod_{i=0}^{r-1} \frac{(2^n-2^i)}{(2^r-2^i)} \le \frac{2^{nr}}{2^{r^2}}$. Thus, by a union bound, we get the following.

$$\Pr_y[\mathrm{rank}(y) = r] \le \frac{2^{nr}}{2^{r^2}} \cdot 2^{-n(n-r)} = 2^{-(n-r)^2} .$$

◁

4 Correlation of random d-linear forms

In this section, we study the correlation of random d-linear forms with lower degree polynomials.

Our main result in this section is the following theorem, which states that a random d-linear form is uncorrelated with degree-ℓ polynomials under certain conditions.

▶ **Theorem 20.** *Let ℓ, d, n be integers such that d divides n, $d = o\left(\frac{n}{\log n}\right)$ and $\ell < d/2$. Set $k = n/d$. Pick a uniformly random d-linear form $f : (\mathbb{F}_2^k)^d \to \mathbb{F}_2$. Then, with probability $1 - o(1)$, f has the following property. For all polynomials $P(X_1, \ldots, X_n) \in \mathbb{F}_2[X_1, \ldots, X_n]$ with degree at most ℓ, we have,*

$$\mathrm{corr}(f, P) < 2^{-(1-o(1))n/d} .$$

Along the way, we develop several tools to understand the bias of random d-linear forms. For example, we show that a random d-linear form is unbiased with extremely high probability.

▶ **Theorem 21.** *Let $\varepsilon > 0$ be fixed. Let d, k be integers with $d < 2^{\varepsilon k/5}$, and consider a uniformly random d-linear form $f : (\mathbb{F}_2^k)^d \to \mathbb{F}_2$. Then,*

$$\Pr\left[\text{bias}(f) \geq 2^{-(1-\varepsilon)k}\right] \leq 2^{-\Omega(\varepsilon^2 k^d)} .$$

▶ Remark 22. Note that any d-linear form $f(X_1, \ldots, X_d)$ vanishes if any one of the block of variables $X_1, \ldots, X_d$ is zero. Hence, the bias of any d-linear form (or equivalently its correlation with the constant 0 polynomial) is at least $2^{-k} = 2^{-n/d}$. Theorem 21 states that it is extremely unlikely for a random d-linear form to have even slightly more bias while Theorem 20 states that it is extremely unlikely for a random d-linear form to have slightly better correlation with any degree ℓ polynomial.

The key ingredient in the proofs of the above theorems is the following theorem on the distribution of the sum of random rank-1 tensors.

▶ **Theorem 23.** *Let $\varepsilon > 0$ be a constant. Let d, k, t be integers with $d < 2^{\varepsilon k/5}$, and $t < \frac{\varepsilon}{5} k^{d-1}$. Let $\{x^{(i,j)}\}_{i\in[t], j\in[d]}$ be picked independently and uniformly distributed in $\mathbb{F}_2^k$. Then,*

$$\Pr\left[\sum_{i=1}^{t} \bigotimes_{j=1}^{d} x^{(i,j)} = 0\right] \leq 2^{-(1-\varepsilon/2)\cdot kt} .$$

▶ Remark 24. If any block of vectors (say wlog. $\{x^{(i,1)}\}_{i\in[t]}$, the first block of vectors) are all 0 (this happens with probability 2^{-kt}), then the d-dimensional linear form $\sum_{i=1}^{t} \bigotimes_{j=1}^{d} x^{(i,j)} = 0$. The above theorem states that the probability of the d-linear form vanishing is not significantly larger.

In turn, the proof of the above theorem is based on the following lemma, which gives an upper bound on the probability that a random rank-1 tensor lies in a fixed low dimensional subspace.

▶ **Lemma 25.** *Let k, d be integers and U be a subspace of $(\mathbb{F}_2^k)^{\otimes d}$ of dimension u. Let $x_1, \ldots, x_d \in \mathbb{F}_2^k$ be picked independently and uniformly at random, and let $T = \otimes_{i=1}^{d} x_i$. Then,*

$$\Pr[T \in U] \leq \frac{d}{2^k} + \frac{2^{u/k^{d-1}}}{2^k} .$$

▶ Remark 26. Let $U = V \otimes (\mathbb{F}_2^k)^{\otimes(d-1)}$ where V is a u/k^{d-1}-dimensional subspace of $\mathbb{F}_2^k$. Note, $\dim(U) = u$. Clearly, $\Pr[\otimes_{i=1}^{d} x_i \in U] = \Pr[x_1 \in V] = 2^{u/k^{d-1}}/2^k$. The above lemma states that the probability is not significantly larger than this for any other U.

In the next subsection, we show how Theorem 20 and Theorem 21 follow from Theorem 23. We defer the proof of Lemma 25 and Theorem 23 to Appendix A.

4.1 Proofs of Theorem 20 and Theorem 21

We first prove Theorem 21.

Proof of Theorem 21. We want to bound $\Pr_f[\text{bias}(f) \geq 2^{-(1-\varepsilon)k}]$. We shall do so by bounding the t^{th} moment of $\text{bias}(f)$ for a suitable choice of t and applying Markov's inequality.

Let $T : [k]^d \to \mathbb{F}_2$ denote the tensor associated with f. Thus $T(i_1, \ldots, i_d)$ are all independent and uniformly distributed in $\mathbb{F}_2$.

We now compute the t^{th} moment of f.

$$\begin{aligned}
&\mathbb{E}_f[(\mathrm{bias}(f))^t] \\
&= \mathbb{E}_f\left[\left(\mathbb{E}_{x^{(1)},\ldots,x^{(d)}\sim\mathbb{F}_2^k}\left[(-1)^{f\left(x^{(1)},\ldots,x^{(d)}\right)}\right]\right)^t\right] \\
&= \mathbb{E}_f\left[\prod_{i\in[t]}\left(\mathbb{E}_{x^{(i,1)},\ldots,x^{(i,d)}\sim\mathbb{F}_2^k}\left[(-1)^{f\left(x^{(i,1)},\ldots,x^{(i,d)}\right)}\right]\right)\right] \\
&= \mathbb{E}_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\mathbb{E}_f\left[(-1)^{\sum_{i=1}^t f\left(x^{(i,1)},\ldots,x^{(i,d)}\right)}\right]\right] \\
&= \mathbb{E}_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\prod_{(\ell_1,\ldots,\ell_d)\in[k]^d}\left(\mathbb{E}_{T(\ell_1,\ldots,\ell_d)\sim\mathbb{F}_2}\left[(-1)^{T(\ell_1,\ldots,\ell_d)\cdot\left(\sum_{i=1}^t\prod_{j=1}^d x_{\ell_j}^{(i,j)}\right)}\right]\right)\right] \\
&= \mathbb{E}_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\prod_{(\ell_1,\ldots,\ell_d)\in[k]^d}\mathbb{1}_{\sum_{i=1}^t\prod_{j=1}^d x_{\ell_j}^{(i,j)}=0}\right] \\
&= \mathbb{E}_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\mathbb{1}_{\forall(\ell_1,\ldots,\ell_d)\in[k]^d,\ \sum_{i=1}^t\prod_{j=1}^d x_{\ell_j}^{(i,j)}=0}\right] \\
&= \Pr_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\forall(\ell_1,\ldots,\ell_d)\in[k]^d,\ \sum_{i=1}^t\prod_{j=1}^d x_{\ell_j}^{(i,j)}=0\right] \\
&= \Pr_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\sum_{i=1}^t\bigotimes_{j=1}^d x^{(i,j)}=0\right].
\end{aligned}$$

Setting $t=\frac{\varepsilon}{10}k^{d-1}$, Theorem 23 tells us that

$$\mathbb{E}_f[(\mathrm{bias}(f))^t]=2^{-(1-\varepsilon/2)kt}.$$

Using Markov's inequality,

$$\Pr_f\left[\mathrm{bias}(f)\geq 2^{-(1-\varepsilon)k}\right]\leq\frac{2^{-(1-\varepsilon/2))kt}}{2^{-(1-\varepsilon)kt}}\leq 2^{-\varepsilon kt/2}\leq 2^{-\Omega(\varepsilon^2k^d)}$$

as claimed. ◀

We now use a similar argument to prove Theorem 20.

Proof of Theorem 20. Fix an arbitrary $\varepsilon>0$. Let $\mathcal{C}$ denote the space of degree $\leq\ell$ polynomials in $\mathbb{F}_2[X_1,\ldots,X_n]$. We want to show that with high probability over the choice of f, we have that for every $P\in\mathcal{C}$, $\mathrm{corr}(f,P)\leq 2^{-(1-\varepsilon)k}$.

Fix $P\in\mathcal{C}$ and consider the t^{th} moment of $\mathrm{bias}(f-P)$. Imitating the proof of Theorem 21, we get

$$\begin{aligned}
&\mathbb{E}_f[(\mathrm{bias}(f-P))^t] \\
&= \mathbb{E}_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[(-1)^{\sum_{i=1}^t P\left(x^{(i,1)},\ldots,x^{(i,d)}\right)}\cdot\mathbb{1}_{\forall(\ell_1,\ldots,\ell_d)\in[k]^d,\ \sum_{i=1}^t\prod_{j=1}^d x_{\ell_j}^{(i,j)}=0}\right] \\
&\leq \mathbb{E}_{\{x^{(i,j)}\}_{i\in[t],j\in[d]}}\left[\mathbb{1}_{\forall(\ell_1,\ldots,\ell_d)\in[k]^d,\ \sum_{i=1}^t\prod_{j=1}^d x_{\ell_j}^{(i,j)}=0}\right] \\
&= \Pr\left[\sum_{i=1}^t\bigotimes_{j=1}^d x^{(i,j)}=0\right].
\end{aligned}$$

Now we will apply Theorem 23. Observe that since $d = o(n/\log n)$, we have,

$$d < 2^{\varepsilon k/5}\,.$$

As in the proof of Theorem 21, we set $t = \frac{\varepsilon}{10}k^{d-1}$, invoke Theorem 23 and apply Markov's inequality to get,

$$\Pr_f\left[\text{bias}(f-P) \geq 2^{-(1-\varepsilon)k}\right] \leq 2^{-\varepsilon^2 k^d/20}\,.$$

Now $\text{bias}(f-P) = \text{corr}(f,P)$. Thus, by a union bound over all $P \in \mathcal{C}$, we have the following.

$$\Pr_f\left[\text{corr}(f,\mathcal{C}) \geq 2^{-(1-\varepsilon)k}\right] \leq |\mathcal{C}| \cdot 2^{-\varepsilon^2 k^d/20}. \tag{4}$$

It remains to estimate $|\mathcal{C}|$. We show below that $|\mathcal{C}| = 2^{o(k^d)}$. The proof of this lemma works for any other C as long as C satisfies $|\mathcal{C}| = 2^{o(k^d)}$. Note that $|\mathcal{C}| = 2^{\binom{n}{\leq \ell}}$. Let δ denote d/n.

$$\begin{aligned}\binom{n}{\leq \ell} \leq \binom{n}{\leq d/2} &\leq \left(\frac{2en}{d}\right)^{d/2} \leq \left(\frac{2e}{\delta}\right)^{\delta n/2} \\ &= o\left(\left(\frac{1}{\delta}\right)^{\delta n}\right) \qquad [\text{Since } \delta = o(1)] \\ &= o(k^d)\,.\end{aligned}$$

Combining this with Equation (4), we get,

$$\Pr_f\left[\text{corr}(f,\mathcal{C}) \geq 2^{-(1-\varepsilon)k}\right] \leq 2^{o(k^d)} \cdot 2^{-\varepsilon^2 k^d/20}\,.$$

Since this holds for every $\varepsilon > 0$, we get the desired result. ◀

References

1. Boris Alexeev, Michael A. Forbes, and Jacob Tsimerman. Tensor rank: Some lower and upper bounds. In *Proc. 26th IEEE Conf. on Comput. Complexity*, pages 283–291, 2011. doi:10.1109/CCC.2011.28.
2. Ido Ben-Eliezer, Rani Hod, and Shachar Lovett. Random low-degree polynomials are hard to approximate. *Comput. Complexity*, 21(1):63–81, 2012. (Preliminary version in *13th RANDOM*, 2009). doi:10.1007/s00037-011-0020-6.
3. Eli Ben-Sasson and Swastik Kopparty. Affine dispersers from subspace polynomials. *SIAM J. Comput.*, 41(4):880–914, 2012. (Preliminary version in *41st STOC*, 2009). doi:10.1137/110826254.
4. Markus Bläser. A $\frac{5}{2}n^2$-lower bound for the rank of $n \times n$ matrix multiplication over arbitrary fields. In *Proc. 40th IEEE Symp. on Foundations of Comp. Science (FOCS)*, pages 45–50, 1999. doi:10.1109/SFFCS.1999.814576.
5. Mark R. Brown and David P. Dobkin. An improved lower bound on polynomial multiplication. *IEEE Trans. Computers*, C-29(5):337–340, 1980. doi:10.1109/TC.1980.1675583.
6. Suryajith Chillara, Mrinal Kumar, Ramprasad Saptharishi, and V. Vinay. The chasm at depth four, and tensor rank : Old results, new insights, 2016. arXiv:1606.04200.
7. David V. Chudnovsky and Gregory V. Chudnovsky. Algebraic complexities and algebraic curves over finite fields. *J. Complexity*, 4(4):285–316, 1988. doi:10.1016/0885-064X(88)90012-X.

8 Klim Efremenko, Ankit Garg, Rafael Mendes de Oliveira, and Avi Wigderson. Barriers for rank methods in arithmetic complexity. In Anna Karlin, editor, *Proc. 9th Innovations in Theor. Comput. Sci. (ITCS)*, volume 94 of *LIPIcs*, pages 1:1–1:19. Schloss Dagstuhl, 2018. arXiv:1710.09502.

9 William Tomothy Gowers and Julia Wolf. Linear forms and higher-degree uniformity for functions on $\mathbb{F}_p^n$. *Geom. Funct. Anal.*, 21(1):36–69, 2011. arXiv:1002.2208.

10 Johan Håstad. Tensor rank is NP-complete. *J. Algorithms*, 11(4):644–654, 1990. (Preliminary version in *16th ICALP*, 1989). doi:10.1016/0196-6774(90)90014-6.

11 Michael Kaminski. A lower bound on the complexity of polynomial multiplication over finite fields. *SIAM J. Comput.*, 34(4):960–992, 2005. (Preliminary version in *22nd STACS*, 2005). doi:10.1137/S0097539704442118.

12 Valentin K. Kolchin. *Random Graphs*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1998. doi:10.1017/CBO9780511721342.

13 Shachar Lovett. The analytic rank of tensors is subadditive, and its applications. *Discrete Analysis*, 2019(7), 2019. arXiv:1806.09179.

14 Robert J. Mceliece, Eugene R. Rodemich, Howard Rumsey, Lloyd, and R. Welch. New upper bounds on the rate of a code via the Delsarte-MacWilliams inequalities. *IEEE Trans. Inform. Theory*, 23(2):157–166, 1977. doi:10.1109/TIT.1977.1055688.

15 Noam Nisan and Avi Wigderson. Hardness vs. randomness. *J. Comput. Syst. Sci.*, 49(2):149–167, October 1994. (Preliminary version in *29th FOCS*, 1988). doi:10.1016/S0022-0000(05)80043-1.

16 Ran Raz. Tensor-rank and lower bounds for arithmetic formulas. *J. ACM*, 60(6):40:1–40:15, 2013. (Preliminary version in *42nd STOC*, 2010). doi:10.1145/2535928.

17 Alexander A. Razborov. Lower bounds on the size of bounded depth circuits over a complete basis with logical addition. *Mathematicheskie Zametki*, 41(4):598–607, 1987. (English translation in *Mathematical Notes of the Academy of Sciences of the USSR*, 41(4):333–338, 1987). doi:10.1007/BF01137685.

18 Igor E. Shparlinski, Michael A. Tsfasman, and Serge G. Vladut. Curves with many points and multiplication in finite fields. In Henning Stichtenoth and Michael A. Tsfasman, editors, *Proc. Int. Workshop on Coding Theory and Algebraic Geometry*, volume 1518 of *LNM*, pages 145–169. Springer, 1992. doi:10.1007/BFb0087999.

19 Amir Shpilka. Lower bounds for matrix product. *SIAM J. Comput.*, 32(5):1185–1200, 2003. (Preliminary version in *42nd FOCS*, 2001). arXiv:cs/0201001.

20 Roman Smolensky. Algebraic methods in the theory of lower bounds for Boolean circuit complexity. In *Proc. 19th ACM Symp. on Theory of Computing (STOC)*, pages 77–82, 1987. doi:10.1145/28395.28404.

21 Volker Strassen. Die berechnungskomplexität von elementarsymmetrischen funktionen und von interpolationskoeffizienten (German) [The computational complexity of elementary symmetric functions and interpolation coefficients]. *Numerische Mathematik*, 20(3):238–251, June 1973. doi:10.1007/BF01436566.

22 Emanuele Viola. On the power of small-depth computation. *Foundations and Trends in Theoretical Computer Science*, 5(1):1–72, 2009. doi:10.1561/0400000033.

23 Emanuele Viola. Matching Smolensky's correlation bound with majority. (manuscript), 2019.

24 Emanuele Viola and Avi Wigderson. Norms, XOR lemmas, and lower bounds for polynomials and protocols. *Theory Comput.*, 4(1):137–168, 2008. (Preliminary version in *22nd CCC*, 2007). doi:10.4086/toc.2008.v004a007.

A Random rank-1 tensors

In this subsection, we first prove Lemma 25 on the probability that a random rank-1 tensor lies in a fixed low-dimensional subspace. We then give a corollary of this lemma which bounds the probability that a collection of random rank-1 tensors spans a very low dimensional subspace. This corollary will be used in the proof of Theorem 23.

Proof of Lemma 25. Define

$$f_{d,k}(u) = \left(1 - \left(1 - \frac{1}{2^k}\right)^{d-1}\right) + \left(1 - \frac{1}{2^k}\right)^{d-1} \cdot \frac{2^{u/k^{d-1}}}{2^k}\,.$$

We will prove, by induction on d, the following stronger bound.

$$\Pr[T \in U] \leq f_{d,k}(u)\,.$$

The fact that this implies the lemma, follows from the observations that $1 - \frac{d-1}{2^k} \leq \left(1 - \frac{1}{2^k}\right)^{d-1}$ and that $\left(1 - \frac{1}{2^k}\right)^{d-1} \leq 1$.

Base case. The $d = 1$ case is trivial (using the observation that $f_{1,k}(u) = \frac{2^u}{2^k}$). We now show the statement holds for larger d.

Induction step. Let $k' = k^{d-1}$. We will view $(\mathbb{F}_2^k)^{\otimes d}$ as $\mathbb{F}_2^k \otimes \mathbb{F}_2^{k'}$. Every element v of $(\mathbb{F}_2^k)^{\otimes d}$ can thus be written as a tuple $(v_1, \ldots, v_k)$, where each v_i is an element of $\mathbb{F}_2^{k'}$ (thus the k^d coordinates are partitioned into k blocks of coordinates, with each block having k' coordinates). We let $\pi_i : (\mathbb{F}_2^k)^{\otimes d} \to \mathbb{F}_2^{k'}$ be the ith projection map, mapping v to v_i.

With this convention, we take a basis for U in *row echelon form*. Concretely, this gives us a basis $\mathcal{B}$ for U, such that $\mathcal{B}$ is a disjoint union of $\mathcal{B}_1, \ldots, \mathcal{B}_k$ ($\mathcal{B}_j$ is the set of basis vectors pivoted in the j'th block of coordinates), such that,

- for all $v \in \mathcal{B}_j$ and $i < j$, $\pi_i(v) = 0$,
- the vectors $\pi_j(v) \in \mathbb{F}_2^{k'}$, as v varies in $\mathcal{B}_j$, are linearly independent.

Define $U_j = \mathrm{span}\{\pi_j(v) \mid v \in \mathcal{B}_j)\}$. Thus we have $\dim(U_j) = |\mathcal{B}_j|$ and

$$\sum_{j=1}^{k} \dim(U_j) = \dim(U)\,.$$

For $i > j$, we define a linear map $\psi_{ij} : U_j \to \mathbb{F}_2^{k'}$ by defining ψ_{ij} on a basis for U_j:

$$\psi_{ij}(\pi_j(v)) = \pi_i(v),\ \forall v \in B_j\,.$$

Then we have the following basic claim (which follows immediately from the above echelon form representation of U).

▷ Claim 27. Let $v \in (\mathbb{F}_2^k)^{\otimes d}$. Then $v \in U$ only if there exists $(u_1, \ldots, u_k) \in \prod_{i=1}^{k} U_i$ such that for each $i \in [k]$ we have

$$\pi_i(v) = u_i + \sum_{j<i} \psi_{ij}(u_j)\,.$$

To simplify notation, we will denote x_1 by y and $\otimes_{i=2}^{d} x_i$ by z. We want to find an upper bound on $\Pr[y \otimes z \in U]$.

▷ Claim 28. Let $\tilde{z} \in (\mathbb{F}_2^k)^{\otimes(d-1)}$ and $S = \{i \mid \tilde{z} \in U_i\}$, then, $\Pr_{y \in \mathbb{F}_2^k}[y \otimes \tilde{z} \in U] \leq \frac{2^{|S|}}{2^k}$.

Proof. For fixed $\tilde{z}$, given the random variable $v = y \otimes \tilde{z}$, we define random variables $u_1, u_2, ..., u_k$ by: $u_i := \pi_i(v) - \sum_{j<i} \psi_{ij}(u_j)$. Note that $\pi_i(v) = \pi_i(y \otimes \tilde{z}) = y_i \tilde{z}$. Also note that u_i is only a function of $y_1, \ldots, y_i$. By Claim 27, $v \in U$ only if for all i, $u_i \in U_i$.

$$
\begin{aligned}
&\Pr_{y \in \mathbb{F}_2^k}[y \otimes \tilde{z} \in U] \\
&\leq \Pr_{y}[\forall i \leq k,\ u_i \in U_i] \\
&= \prod_{i=1}^{k} \Pr[u_i \in U_i \mid u_1 \in U_1, \ldots, u_{i-1} \in U_{i-1}] \\
&= \prod_{i=1}^{k} \mathbb{E}_{u_1 \in U_1, \ldots, u_{i-1} \in U_{i-1}} \left[\Pr_{u_i}[u_i \in U_i \mid u_1, \ldots, u_{i-1}] \right] \\
&= \prod_{i=1}^{k} \mathbb{E}_{u_1 \in U_1, \ldots, u_{i-1} \in U_{i-1}} \left[Pr_{u_i} \left[\pi_i(v) - \sum_{j<i} \psi_{ij}(u_j) \in U_i \,\middle|\, u_1, \ldots, u_{i-1} \right] \right] \\
&= \prod_{i=1}^{k} \mathbb{E}_{u_1 \in U_1, \ldots, u_{i-1} \in U_{i-1}} \left[Pr_{y_i} \left[y_i \tilde{z} - \sum_{j<i} \psi_{ij}(u_j) \in U_i \,\middle|\, u_1, \ldots, u_{i-1} \right] \right] \\
&\leq \prod_{i \notin S} \left(\frac{1}{2} \right) = \left(\frac{1}{2} \right)^{k-|S|},
\end{aligned}
$$

where the last inequality follows since for every $i \notin S$ and every vector w, at most one of w and $w + \tilde{z}$ can lie in U_i (as $\tilde{z} \notin U_i$). ◁

For $S \subseteq [k]$, let $U_S = \bigcap_{i \in S} U_i$. Then,

$$
\begin{aligned}
\Pr_{y,z}[y \otimes z \in U] &\leq \mathbb{E}_z \left[\frac{2^{\sum_{i=1}^{k} 1_{U_i}(z)}}{2^k} \right] \qquad \text{[Follows from the above claim]} \\
&= \frac{1}{2^k} \mathbb{E}_z \left[\prod_{i=1}^{k} 2^{1_{U_i}(z)} \right] \\
&= \frac{1}{2^k} \mathbb{E}_z \left[\prod_{i=1}^{k} (1 + 1_{U_i}(z)) \right] \\
&= \frac{1}{2^k} \mathbb{E}_z \left[\sum_{S \subseteq [k]} 1_{U_S}(z) \right] \\
&= \frac{1}{2^k} \sum_{S \subseteq [k]} \Pr_z[z \in U_S] \,.
\end{aligned}
$$

Now, observe that for each $i \in S$, we have $\Pr[z \in U_S] \leq \Pr[z \in U_i]$. Thus if we sort the U_i so that $\dim(U_1) \geq \dim(U_2) \geq \ldots \geq \dim(U_k)$, then we have the following sequence of inequalities.

$$\begin{aligned}\Pr_{y,z}[y \otimes z \in U] &\leq \frac{1}{2^k}\left(1 + \sum_{i\in[k]} \sum_{S\subseteq[i], i\in S} \Pr_z[z \in U_S]\right) \\ &\leq \frac{1}{2^k}\left(1 + \sum_{i\in[k]} 2^{i-1} \Pr_z[z \in U_i]\right) \\ &\leq \frac{1}{2^k}\left(1 + \sum_{i\in[k]} 2^{i-1} f_{d-1,k}(\dim(U_i))\right),\end{aligned}$$

where the last step follows from the induction hypothesis. To find an upper bound for this last expression, we let $a_i = \dim(U_i)$. We have the constraints

$$\sum_i a_i = u\,,$$

$$k' \geq a_1 \geq a_2 \geq \ldots \geq a_k \geq 0\,,$$

where $k' = k^{d-1}$, and we want to maximize an expression of the form

$$\sum_{i=1}^{k} 2^{i-1}(\alpha + \beta 2^{a_i/k^{d-2}}) = \alpha \cdot (2^k - 1) + \beta \cdot \left(\sum_{i=1}^{k} 2^{i-1+a_i/k^{d-2}}\right).$$

where $\alpha, \beta > 0$.

It is worth noting what happens in the two examples $U = V \otimes \mathbb{F}_2^{k'}$ and $U = \mathbb{F}_2^k \otimes W$, where $V \subseteq \mathbb{F}_2^k$ and $W \subseteq \mathbb{F}_2^{k'}$ are subspaces of the appropriate dimension. In the first case, $a_1 = a_2 = \ldots = a_{u/k'} = k'$ and the remaining a_i are 0. In the second case, all the $a_i = u/k$. Both are global maxima of the expression we want to maximize! The existence of these very different maxima makes this maximization problem somewhat tricky.

In Theorem 29 we prove a tight upper bound for this function. For every $i \in [k]$, let $b_i = a_i/k^{d-2}$, and let $\tilde{u} = u/k^{d-2}$. Then, $b_1, b_2, \ldots, b_k$ and $\tilde{u}$ satisfy the constraints in the hypothesis of Theorem 29, and Theorem 29 tells us that a global maxima is achieved when all the a_i are equal to $\dim(U)/k$. Thus,

$$\begin{aligned}\Pr_{y,z}[y \otimes z \in U] &\leq \frac{1}{2^k}\left(1 + \sum_{i\in[k]} 2^{i-1} f_{d-1,k}(u/k)\right) \\ &= \frac{1}{2^k}\left(1 + (2^k - 1) f_{d-1,k}(u/k)\right) \\ &= \left(\frac{1}{2^k} + (1 - \frac{1}{2^k}) f_{d-1,k}(u/k)\right) \\ &= f_{d,k}(u)\,.\end{aligned}$$

This completes the induction step. ◀

▶ **Theorem 29.** *Let k be a positive integer, and let $\tilde{u} \in [0, k^2]$ be a real number. Suppose $b_1, b_2, \ldots, b_k$ are real numbers satisfying the following constraints.*

$$k \geq b_1 \geq b_2 \ldots \geq b_k \geq 0\,, \tag{5}$$

$$\sum_{i=1}^{k} b_i = \tilde{u}\,. \tag{6}$$

Then,

$$\sum_{i=1}^{k} 2^{i-1} 2^{b_i} \leq \sum_{i=1}^{k} 2^{i-1} 2^{\tilde{u}/k} = (2^k - 1) 2^{\tilde{u}/k} .$$

We refer the reader to the full version of our paper for a proof of Theorem 29.

We now use the previous lemma to prove a corollary about the dimension of the span of several random rank 1 tensors.

▶ **Corollary 30.** *Let d, k, t be integers. For each $i \in [t]$ and $j \in [d]$, pick $x^{(i,j)} \in \mathbb{F}_2^k$ uniformly at random. For $i \in [t]$, let T_i be the rank-1 tensor $\otimes_{j=1}^{d} x^{(i,j)}$. Then, for every $0 \leq r \leq t$,*

$$\Pr[\dim(\mathrm{span}(\{T_1, \ldots, T_t\})) = r] \leq \binom{t}{r} \left(\frac{d + 2^{t/k^{d-1}}}{2^k} \right)^{t-r} .$$

Proof. Let us reveal $T_1, \ldots, T_t$ one at a time. For $0 \leq i \leq t$, let $V_i = \mathrm{span}(\{T_1, \ldots, T_{i-1}, T_i\})$. Thus we have $0 = \dim(V_0) \leq \dim(V_1) \leq \ldots \dim(V_t)$. We want to estimate the probability that $\dim(V_t) = r$. Let E_i denote the event that $T_i \in V_{i-1}$. For $I \subseteq [t]$, let E_I denote the event $\bigcap_{i \in I} E_i$. In terms of these events, we can bound $\Pr[\dim(V_t) = r]$ as follows.

$$\begin{aligned} \Pr[\dim(V_t) = r] &\leq \Pr[\exists I \subseteq [t], |I| = t - r \text{ such that } E_I \text{ occurs}] \\ &\leq \sum_{I \subseteq [t], |I| = t-r} \Pr[E_I] . \end{aligned}$$

We conclude the proof by bounding $\Pr[E_I]$. Fix $I \subseteq [t]$ with $|I| = t - r$. Let $I = \{i_1, \ldots, i_{t-r}\}$ with $i_1 < i_2 < \ldots < i_{t-r}$.

$$\Pr[E_I] = \prod_{j=1}^{t-r} \Pr[E_{i_j} | \bigcap_{\ell < j} E_{i_\ell}] .$$

Lemma 25 implies the following.

$$\Pr[E_i | T_1, \ldots, T_{i-1}] \leq \frac{d + 2^{\dim(V_{i-1})/k^{d-1}}}{2^k} .$$

For any given $j \in [t - r]$, the events $E_{i_1}, \ldots, E_{i_{j-1}}$ are all determined by $T_1, \ldots, T_{i_j - 1}$ (since E_{i_ℓ} depends on $T_1, \ldots, T_{i_\ell}$, and $i_{j-1} \leq i_j - 1$). Thus, for each $j \in [t - r]$, we have,

$$\Pr[E_{i_j} | \bigcap_{\ell < j} E_{i_\ell}] \leq \frac{d + 2^{t/k^{d-1}}}{2^k} .$$

Here we used the fact that $\dim(V_{i_j - 1}) \leq t$. Using this in our previous bound, we conclude that

$$\Pr[E_I] \leq \left(\frac{d + 2^{t/k^{d-1}}}{2^k} \right)^{t-r} ,$$

and thus,

$$\Pr[\dim(V_t) = r] \leq \binom{t}{r} \cdot \left(\frac{d + 2^{t/k^{d-1}}}{2^k} \right)^{t-r} . \qquad \blacktriangleleft$$

A.1 Proof of Theorem 23

We now use Corollary 30 to prove Theorem 23.

Proof of Theorem 23. The equation

$$\sum_{i=1}^{t} \bigotimes_{j=1}^{d} x^{(i,j)} = 0 \tag{7}$$

implies that

$$\forall \ell \in [k],\ \sum_{i=1}^{t} x_{\ell}^{(i,1)} \cdot \bigotimes_{j=2}^{d} x^{(i,j)} = 0\,. \tag{8}$$

Let T_i denote $\bigotimes_{j=2}^{d} x^{(i,j)}$ for $i \in [t]$ and $\mathcal{T} = \mathrm{span}(\{T_1, \ldots, T_t\})$. Then we have,

$$\begin{aligned}
&\Pr[\{x^{(i,j)}\}_{i\in[t],j\in[d]} \text{ satisfy (7)}] \qquad (9)\\
&\leq \Pr[\{x^{(i,j)}\}_{i\in[t],j\in[d]} \text{ satisfy (8)}]\\
&= \sum_{r=0}^{t} \Pr\left[\{x^{(i,j)}\}_{i\in[t],j\in[d]} \text{ satisfy (8)} \middle| \dim(\mathcal{T}) = r\right] \Pr\left[\dim(\mathcal{T}) = r\right]\\
&= \sum_{r=0}^{t} \left(\prod_{\ell\in[k]} \Pr\left[\sum_{i=1}^{t} x_{\ell}^{(i,1)} \cdot T_i = 0 \middle| \dim(\mathcal{T}) = r \right] \right) \cdot \Pr\left[\dim(\mathcal{T}) = r\right]\\
&\leq \sum_{r=0}^{t} \left(\frac{1}{2^r}\right)^k \cdot \Pr\left[\dim(\mathcal{T}) = r\right]. \qquad (10)
\end{aligned}$$

Here, the equality in the third step follows from the fact that $\{x_{\ell}^{(i,1)}\}_{i\in[t],\ell\in[k]}$ are independently and uniformly distributed in $\mathbb{F}_2$.
By the given distribution of $T_1, \ldots, T_t$ in $(\mathbb{F}_2^k)^{\otimes(d-1)}$, Corollary 30 says that

$$\Pr\left[\dim(\mathcal{T}) = r\right] \leq \binom{t}{r} \left(\frac{d - 1 + 2^{\frac{t}{k^{d-2}}}}{2^k}\right)^{t-r}.$$

Plugging this bound back into (9) gives

$$\begin{aligned}
\Pr[\{x^{(i,j)}\}_{i\in[t],j\in[d]} \text{ satisfy (7)}] &\leq \sum_{r=0}^{t} \binom{t}{r} \frac{1}{2^{rk}} \left(\frac{d - 1 + 2^{\frac{t}{k^{d-2}}}}{2^k}\right)^{t-r}\\
&\leq \sum_{r=0}^{t} \binom{t}{r} \left(\frac{1}{2^k}\right)^r \left(\frac{d - 1 + 2^{\frac{t}{k^{d-2}}}}{2^k}\right)^{t-r}\\
&= \left(\frac{1}{2^k} + \frac{d - 1 + 2^{\frac{t}{k^{d-2}}}}{2^k}\right)^t\\
&\leq \left(\frac{d + 2^{\frac{t}{k^{d-2}}}}{2^k}\right)^t.
\end{aligned}$$

Now, since $d < 2^{\varepsilon k/5}$ and $t < \varepsilon k^{d-1}/5$, we have

$$d + 2^{\frac{t}{k^{d-2}}} < 2 \cdot 2^{\varepsilon k/5} < 2^{\varepsilon k/2},$$

we conclude that

$$\Pr\left[\sum_{i=1}^{t} \bigotimes_{j=1}^{d} x^{(i,j)} = 0\right] < 2^{-(1-\varepsilon/2)kt}.$$

This completes the proof. ◀

On the List Recoverability of Randomly Punctured Codes

Ben Lund
Department of Mathematics, Princeton University, NJ, USA
http://www.ben-lund.com
lund.ben@gmail.com

Aditya Potukuchi
Department of Computer Science, Rutgers University, Piscataway, NJ, USA
https://www.adityapotukuchi.com
aditya.potukuchi@cs.rutgers.edu

Abstract

We show that a random puncturing of a code with good distance is list recoverable beyond the Johnson bound. In particular, this implies that there are Reed-Solomon codes that are list recoverable beyond the Johnson bound. It was previously known that there are Reed-Solomon codes that do not have this property. As an immediate corollary to our main theorem, we obtain better degree bounds on unbalanced expanders that come from Reed-Solomon codes.

2012 ACM Subject Classification Theory of computation → Error-correcting codes

Keywords and phrases List recovery, randomly punctured codes, Reed-Solomon codes

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.30

Category RANDOM

Funding *Ben Lund*: Research supported by NSF grant DMS-1802787.
Aditya Potukuchi: Research supported in part by Noga Ron-Zewi's BSF-NSF grant CCF-1814629 and 2017732, by NSF grant CCF-1350572 and the Simons Collaboration on Algorithms and Geometry.

Acknowledgements We thank Venkat Guruswami, Jeff Kahn, Swastik Kopparty, Noga Ron-Zewi, and Mary Wootters for helpful discussions. We also thank an anonymous reviewer for helpful comments on the presentation of the results.

1 Introduction

List recoverable codes were defined by Guruswami and Indyk [7] in their study of list decodable codes. Here, we study list recoverable codes in their own right, showing that random puncturings of codes over a sufficiently large alphabet are list recoverable. Our result is analogous to earlier work by Rudra and Wooters [10, 11] on the list decodability of randomly punctured codes.

We use q to denote the alphabet size, and n to denote the block length of an arbitrary code $\mathcal{C} \subset [q]^n$. Given two codewords $c_1, c_2 \in [q]^n$, denote the Hamming distance between c_1 and c_2 by $\Delta(c_1, c_2)$. Denote the minimum distance between a codeword $c \in [q]^n$ and a set $\mathcal{L} \subseteq [q]^n$ by $\Delta(c, \mathcal{L})$.

▶ **Definition 1** (List recoverability). *Let q, n, ℓ, L be positive integers, and let $0 \leq \rho < 1$ be a real number. A code $\mathcal{C} \subset [q]^n$ is (ρ, ℓ, L) list recoverable if, for every collection of sets $\{A_i \subseteq [q]\}_{i \in [n]}$ with $|A_i| \leq \ell$ for each i, we have*

$$|\{c \in \mathcal{C} \mid \Delta(c, A_1 \times \cdots \times A_n) \leq \rho n\}| \leq L.$$


© Ben Lund and Aditya Potukuchi;
licensed under Creative Commons License CC-BY

In the above definition, ℓ is called the *input list size*, and L is called the *output list size* from which the code can be recovered. The case $\rho = 0$ is already interesting, and is called *zero-error* list recoverability. We say that a code $\mathcal{C}$ is (ℓ, L) zero-error list recoverable if it is $(0, \ell, L)$ list recoverable.

A *puncturing* of a code $\mathcal{C} \subset [q]^n$ to a set $S \subset [n]$ is the code $\mathcal{C}_S \subset [q]^S$ defined by $\mathcal{C}_S[i] = \mathcal{C}[i]$ for each $i \in S$. A punctured code will typically have higher rate, but lower distance, than the unpunctured version. Our main result is that every code over a large enough alphabet $[q]$ can be punctured to a code of rate $R > q^{-1/2}$ while being list recoverable with input and output list sizes roughly R^{-2}.

▶ **Theorem 2.** *For any real α with $0 < \alpha \leq 1$, every code with distance at least $n(1-q^{-1}-\epsilon^2)$ can be punctured to rate $\Omega\left(\frac{\epsilon}{\log q}\right)$ so that it is $(\rho, \ell, \ell(1+\alpha))$-list recoverable, provided that the following inequalities are satisfied:*

- *n and q are sufficiently large,*
- *$0 \leq \rho < 1 - (1+\alpha)^{-1/2}$,*
- *$q^{-1/2} < \epsilon < \min(c, 2^{-1}\gamma\sigma)$,*
- *$\ell \leq \epsilon^{-2}\sigma^2\gamma$,*

where $c > 0$ is a constant, $\gamma = (1+\alpha)(1-\rho)^2 - 1$, and $\sigma = (1-\rho)(2-\rho)^{-1}$.

In fact, we show a random puncturing is list recoverable with the same list size with high probability; see Theorem 10 for a precise statement.

It is helpful to consider the case $\rho = 0$ and $\alpha = 1$. Then we obtain a $(\ell, 2\ell)$ zero-error list-recoverable code. If we start with a code having distance nearly as large as possible (*i.e.* take $\epsilon = \Theta(q^{-1/2})$), then we obtain input and output list sizes of $\Omega(q)$ and rate $\Omega(q^{-1/2}\log^{-1} q)$. In this way, we obtain a punctured code of rate $\Omega(\ell^{-1/2}\log^{-1}\ell)$. The main point is that this is better than the codes guaranteed by the Johnson bound (discussed in more detail below), which gives a code of rate $\Omega(\ell^{-1})$. A completely random code, however, can be $(\ell, 2\ell)$ zero-error list-recoverable with rate $\Omega(1)$.

One nice feature Theorem 2 is that it can yield an input list size as large as $\Omega(q)$. A second nice feature is that the theorem yields a tight relationship between the input and output list sizes; many results on list recovery give an output list that is much larger than the input list. Finally we mention that the proof of Theorem 2 is relatively simple and elementary; we draw a slightly more explicit comparison to the earlier work of Rudra and Wooters [10, 11] below.

2 Background

In this section, we give a brief discussion about the current state of the literature. Theorem 2 is analogous to a theorem of Rudra and Wooters [10, 11] on the *list decodability* of punctured codes over large alphabets. A code $\mathcal{C} \subset [q]^n$ is (ρ, ℓ)-list decodable if for each $x \in [q]^n$, there are at most ℓ codewords of $\mathcal{C}$ that differ from x in fewer than ρn coordinates.

▶ **Theorem 3** ([11])**.** *Let $\epsilon > q^{-1/2}$ be a real number, and q, n be sufficiently large integers. Every code $\mathcal{C} \subset [q]^n$ with distance $n(1-q^{-1}-\epsilon^2)$ can be punctured to rate $\tilde{\Omega}\left(\frac{\epsilon}{\log q}\right)$ so that it is $(1-O(\epsilon), O(\epsilon^{-1}))$-list decodable.*

Theorems 2 and 3 are most interesting in the case of Reed-Solomon codes. The codewords of the degree-d Reed-Solomon code over $\mathbb{F}_q$ with evaluation set $S \in \binom{[q]}{n}$ are the evaluations of all univariate polynomials of degree at most d on elements of S. In other words, suppose $S = \{s_1, \ldots, s_n\}$. The degree-d Reed-Solomon code on S is the set

$$\{(p(s_1), \ldots, p(s_n)) \mid \deg(p) \leq d\}.$$

The block length of this code is $n \leq q$. Since two distinct polynomials of degree at most d can agree on at most d locations, the distance of any degree-d Reed-Solomon code is at least $n - d$.

A fundamental result, which gives a lower bound on the list decodability of a code with given distance, is the *Johnson bound* (see, for example Corollary 3.2 in [6]).

▶ **Theorem 4** (Johnson bound for list decoding). *Every code $\mathcal{C} \subset [q]^n$ of minimum distance at least $n(1 - q^{-1} - \epsilon^2)$ is $((1 - q^{-1} - \epsilon), O(\epsilon^{-1}))$- list decodable.*

One of the main points of Theorem 3 is that it shows that there are Reed-Solomon codes that are list decodable beyond the Johnson bound. This is a very interesting result because variations of Reed-Solomon codes have been shown to beat the Johnson bound. On the other hand, Reed-Solomon codes are quite natural and some applications in Complexity theory specifically needed Reed-Solomon codes (and not the aforementioned variants).

A similar result as Theorem 4, using a similar argument, also known as the Johnson bound, is known for list recoverability (see for example, Lemma 5.2 in [4]).

▶ **Theorem 5** (Johnson bound for list recovery). *Let $\mathcal{C} \subseteq [q]^n$ be a code of relative distance $1 - \epsilon$. Then $\mathcal{C}$ is (ρ, ℓ, L)-list recoverable for any $\ell \leq \epsilon^{-1}(1 - \rho)^2$ with $L = \frac{\ell}{(1-\rho)^2 - \epsilon\ell}$.*

For perspective, the goal here is to have the input list size ℓ as large as possible for a given ρ while ensuring that the output list size L is small, for example, $L = \text{poly}(\ell)$ is a reasonable regime of efficiency. Theorem 5 roughly says that every q-ary code of distance $1 - q^{-1} - \epsilon$ is $(\rho, \ell, O(\ell))$-list recoverable for $\ell = O(\epsilon^{-1})$ and $\rho \leq 1 - \sqrt{2\epsilon\ell}$. Thus, a question naturally arises:

▶ **Question.** *Are there q-ary Reed-Solomon codes of distance $1 - q^{-1} - \epsilon$ which are (ρ, ℓ, L)-list recoverable for some $\rho \in [0, 1)$ and $\ell = \omega(\epsilon^{-1})$ and $L = \text{poly}(\ell)$?*

We would like to remark again, that the case $\rho = 0$ is also interesting. A result of Guruswami and Rudra [8] shows that there are Reed-Solomon codes where one cannot hope to beat the Johnson bound in the case when $\rho = 0$.

▶ **Theorem 6** (Theorem 1 in [8]). *Let $q = p^m$ where p is a prime, and let $\mathcal{C}$ denote the degree-$\left(\frac{p^m - 1}{p - 1}\right)$ Reed-Solomon code over $\mathbb{F}_q$ with $\mathbb{F}_q$ as the evaluation set. Then there are lists $S_1, \ldots, S_q$ each of size p such that*

$$|\mathcal{C} \cap (S_1 \times \cdots \times S_q)| = q^{2^m}.$$

In the proof of the above theorem, each list A_i is given by $\mathbb{F}_p \subset \mathbb{F}_q$, and the set of codewords contained in these lists are codewords of the *BCH code*. This result exploits the specific subfield structure of the input lists. To understand this result quantitatively, recall that a degree-d Reed-Solomon code has relative distance $1 - \frac{1}{q} - \frac{d}{q}$. Setting $\ell = p - 1$ and $\rho = 0$ in the Johnson bound tells us that such a code is $(p - 1, O(q))$ zero-error list recoverable. Setting the list size as p in the bound gives us nothing, and Theorem 6 says that the number of codewords grows superpolynomially in q.

One thing to note in Theorem 6 is that the same lists of size p also support at least q^{2^m} codewords for any punctured code. Thus in some sense, the reason why Theorem 2 is true is that lists of size $p-1$ do not support many more codewords in an appropriately punctured code than in the un-punctured case. This is very similar to the intuition in the results of [10], [11] for list-decodability.

Theorem 2 immediately gives the following corollary.

▶ **Corollary 7.** *For a prime power q and small enough $\epsilon \geq q^{-1/2}$, there are Reed-Solomon codes of rate $\tilde{\Omega}\left(\frac{\epsilon}{\log q}\right)$ which are $(\epsilon^{-2}, O(\epsilon^{-2}))$ zero-error list recoverable.*

Thus, one is able to go beyond the $O(\epsilon^{-1})$ size input lists. Again, one can easily check that setting $\ell = \epsilon^{-2}$ in the Johnson bound gives nothing. A point worth noting is that there is no inherent upper bound on the input list size, and the input lists can be as large as $\Omega(q)$ when ϵ is around $q^{-1/2}$. In fact, $\ell = \Omega(q)$ and $L \leq 2\ell$ is the regime for the main application of Theorem 2 (see Section 2.2).

A natural attempt at Theorem 2 is to use the method from the aforementioned result of Rudra and Wootters. This method uses tools from high dimensional probability. At a *very* high level, the main intuition in their argument is the following: Suppose there is a set Λ of $L = \Omega(\epsilon^{-1})$ codewords which are at a distance at most $1 - O(\epsilon)$ from some point $x \in [q]^n$, then there is a subset $\Lambda' \subset \Lambda$ which is much smaller where the distribution of distances from x is similar to that of Λ. The existence of such a Λ is a bad event, which is witnessed by a smaller bad event (i.e., existence of Λ') of the *same type*. Thus, this requires one to union bound just over Λ''s which are much smaller in number. Attempting to use this idea in a straightforward way to list recovery seems to be very lossy. The proof of Theorem 2 builds on this idea, and uses the fact that bad events can be witnessed by smaller bad events of a *different* (although related) type (see Section 3.1 for a somewhat more detailed sketch of the proof). Surprisingly though, the execution of this idea in this case is far simpler than in the previous works, and is completely elementary.

2.1 A quantitative summary of rate bounds for list recovery

We summarize the above discussion into a perspective with which one may view Theorem 2. Fix a $\rho \in [0, 1)$ to be the fraction of errors from which we wish to list-recover q-ary codes where q is larger than the block length. Suppose one wanted to list recover from input lists of size ℓ where the output list is $\text{poly}(\ell)$[1], the Johnson bound (Theorem 5) guarantees that Reed-Solomon codes of rate $\Omega(1/\ell)$ achieve this. Theorem 6 says that in general, this dependence cannot be improved, i.e., there are Reed-Solomon codes of rate $O(1/\ell)$ where the output list is superpolynomial for infinitely many q and ℓ. However, Theorem 2 says that most Reed-Solomon codes of rate $\Omega(1/(\ell^{1/2} \log q))$ achieve this.

It should be worth noting that decoding from lists of size ℓ can be achieved by random codes of rate $\Omega(1)$, whereas random Reed-Solomon codes require that the rate is $O(1/\log \ell)$ (Theorem 11). In fact, nothing better is known even for random linear codes. In this sense random codes are much better at list recovery than random Reed-Solomon codes.

[1] The proof of Theorem 2 relies on a "birthday paradox" type argument that cannot exploit the additional structure when one allows L to grow as a larger function of ℓ.

2.2 Unbalanced expander graphs from codes

The zero-error case of Theorem 2 leads to some progress on a question of Guruswami regarding unbalanced expanders obtained from Reed-Solomon graphs. This was also the main motivation behind this theorem.

Informally, an expander graph is a graph where every small set of vertices has a relatively large neighborhood. In this case, we say that all small sets *expand*. One interesting type of expander graphs are *unbalanced expanders*. These are bipartite graphs where one side is much larger than the other side, and we want that all the small subsets of the *larger* side expand.

▶ **Definition 8** (Unbalanced expander). *A (k, d, ϵ)-regular unbalanced expander is a bipartite graph on vertex set $L \sqcup R$, $|L| \geq |R|$ where the degree of every vertex in L is d, and for every $S \subseteq L$ such that $|S| = k$, we have that $|N(S)| \geq d|S|(1-\epsilon)$.*

Note that in the above definition, $|N(S)| \leq d|S|$. We are typically interested in infinite families of unbalanced expanders for which $\epsilon = o(1)$, $d = o(|R|)$, and $k = \tilde{\Omega}(|R|/d)$.

Given a code $\mathcal{C} \subseteq [q]^n$, it is natural to look at the bipartite graph, which we will denote by $G(\mathcal{C})$ where the vertex sets are $\mathcal{C} \sqcup ([n] \times [q])$. For every $c = (c_1, \ldots, c_n) \in \mathcal{C}$ the set of neighbors is $\{(1, c_1), \ldots, (n, c_n)\}$. This graph is especially interesting when $\mathcal{C}$ is a low-degree Reed-Solomon code evaluated at an appropriate set.

The following is a open question in the study of pseudorandomness that is attributed to Guruswami [5], (also explicitly stated in [2]): Fix an integer d. For a subset $S \in \binom{[q]}{m}$, define $\mathcal{C}_S$ to be the degree-d Reed-Solomon code with S as the evaluation set, where d is a constant.

▶ **Question.** *What is the smallest m such that when S is chosen uniformly at random, $G(\mathcal{C}_S)$ is, with high probability, a $(\Omega(q), 1/2)$-unbalanced expander?*

There are examples of explicit constructions unbalanced expanders that come from other means [9]. In fact, [9] also contains a construction based on a variant of Reed-Solomon code known as *folded Reed-Solomon code*. However, the above question has a very natural geometric/combinatorial "core" which is interesting in its own right and so far, seems to evade known techniques.

It was probably well known that $m = \Omega(\log q)$, and we also give a proof of this (Theorem 11) since we could not find it in the literature. But for upper bounds, it seems nothing better than the almost trivial $m = O(q)$ was known [1]. Since the zero-error list recoverability of $\mathcal{C}$ is equivalent to the expansion of $G(\mathcal{C})$, an immediate Corollary to Theorem 10 gives an improved upper bound.

▶ **Corollary 9.** *Let q, n be sufficiently large integers and $\alpha \in (0, 1)$, $\epsilon > q^{-1/2}$ be real numbers. For every code $\mathcal{C} \subset [q]^n$ with relative distance $1 - q^{-1} - \epsilon^2$, there is a subset $S \subseteq [n]$ such that $|S| = O(\epsilon n \log q)$ such that $G(\mathcal{C}_S)$ is a $(\alpha\epsilon^{-2}, |S|, \alpha)$-unbalanced expander.*

Instantiating the above theorem for degree-d Reed-Solomon codes, we have $n = q$ and $\epsilon = (d/q)^{-\frac{1}{2}}$. This gives, $m = \tilde{O}(\sqrt{q})$.

3 Proof of Theorem 2

The bulk of this section is the statement and proof of Theorem 10. After the proof of Theorem 10, we show how to derive Theorem 2 from it.

3.1 A sketch of the proof

Here, we sketch the proof when $\rho = 0$, i.e., for *zero-error* list recovery. This contains most of the main ideas required for the general theorem. Let $S = \{s_1, \ldots, s_m\} \subseteq [n]$ be a randomly chosen evaluation set. The main observation is that if there are input lists $A_1, \ldots, A_m \subseteq [q]$, such that $(A_1 \times \cdots \times A_m)$ contains a large subset $\mathcal{D} \subseteq \mathcal{C}$ of codewords, then there is a small subset $\mathcal{C}' \subseteq \mathcal{D} \subseteq \mathcal{C}$ which agree on an unusually high number of coordinates. An appropriately sized random subset of $\mathcal{D}$ does this. Thus the event that a given puncturing is bad is contained witnessed by the event that there are few codewords that agree a lot on the coordinates chosen in S. The number of events of the latter kind are far fewer in number, leaving us with fewer bad events to overcome for the union bound.

3.2 Proof of Theorem 2

The calculations in the proof of Theorem 10 are all explicit, but we have not tried to optimize the constant terms.

▶ **Theorem 10.** *Let $0 < \alpha < 1$ and $0 \le \rho < 1 - (1+\alpha)^{-1/2}$ be real numbers. Let q, n, d, ℓ, and m be positive integers. Let $\mathcal{C} \subset [q]^n$ be a code of distance at least $n - nq^{-1} - d$. Denote $\gamma = (1+\alpha)(1-\rho)^2 - 1$ and $\sigma = (1-\rho)(2-\rho)^{-1}$. Suppose that the following inequalities are satisfied:*

$$\begin{aligned} d &\ge nq^{-1}, \\ 4\gamma^{-1} \le \ell &\le 800^{-1}\,\sigma\gamma n d^{-1}, \\ \sigma m &\ge 1280\sqrt{\ell\gamma^{-1}}\log|\mathcal{C}|, \\ m &< n. \end{aligned}$$

Then, for $S \in \binom{[n]}{m}$ chosen uniformly at random, the probability that $\mathcal{C}_S$ is $(\rho, \ell, \ell(1+\alpha))$-list recoverable is at least $1 - e^{-\sigma m/64}$.

Proof. For any $\mathcal{C}' \subseteq \mathcal{C}$, denote by $T(\mathcal{C}')$ the set of coordinates $i \in [n]$ such that there is a pair $c_1, c_2 \in \mathcal{C}'$ with $c_1[i] = c_2[i]$.

The basic outline of the proof is to first show that, for any S such that $\mathcal{C}_S$ is not $(\rho, \ell, \ell(1+\alpha))$-list recoverable, there is a pair $S', \mathcal{C}'$ such that S' is large and $|T(\mathcal{C}') \cap S'|$ is unusually large. Taking a union bound over all candidates for $\mathcal{C}'$ then shows that there cannot be too many pairs of this sort.

Let $S \in \binom{[n]}{m}$ so that $\mathcal{C}_S$ is not $(\rho, \ell, \ell(1+\alpha))$-list recoverable. We will show that there is a set $\mathcal{C}' \subset \mathcal{C}_S$ such that

$$|\mathcal{C}'| \le 10\sqrt{\ell/\gamma}, \text{ and} \tag{1}$$

$$|T(\mathcal{C}') \cap S| \ge \sigma m/4. \tag{2}$$

Since $\mathcal{C}_S$ is not $(\rho, \ell, \ell(1+\alpha))$-list recoverable, there are subsets $A_i \subseteq [q]$ for each $i \in S$ such that each $|A_i| \le \ell$ and $|\{c \in \mathcal{C}_S : \Delta(c, \prod_{i\in S} A_i) \le \rho n\}| > \ell(1+\alpha)$.

Let

$$\mathcal{D} = \{c \in \mathcal{C}_s : \Delta(c, \prod_{i\in S} A_i) \le \rho n\}.$$

For $i \in S$, let

$$\mathcal{D}_i = \{c \in \mathcal{D} : c[i] \in A_i\}.$$

Let

$$I = \{(c, i) \in \mathcal{D} \times S : c \in \mathcal{D}_i\}.$$

From the definition of $\mathcal{D}$, we have

$$|I| \geq |\mathcal{D}|(1-\rho)m. \tag{3}$$

Note that the average cardinality of the $\mathcal{D}_i$ is $(1-\rho)|\mathcal{D}|$. Let

$$S' = \{i \in S : |\mathcal{D}_i| \geq (1-\rho)^2|\mathcal{D}|\}.$$

If $\rho = 0$, then $\mathcal{D}_i = \mathcal{D}$ for each i, and hence $|S'| = m$. Next we show that, if $\rho > 0$, then $|S'| \geq (1-\rho)(2-\rho)^{-1}m = \sigma m$. Since $|\mathcal{D}_i \leq |\mathcal{D}|$ for each i, we have

$$|S'|\,|\mathcal{D}| \geq \sum_{i \in S'} |D_i| = |I| - \sum_{i \in S \setminus S'} \mathcal{D}_i. \tag{4}$$

Since $|D_i| < (1-\rho)^2|\mathcal{D}|$ for each $i \in S \setminus S'$, we have

$$\sum_{i \in S \setminus S'} \leq (m - |S'|)(1-\rho)^2|\mathcal{D}|. \tag{5}$$

A straightforward rearrangement of (3), (4), and (5) using the assumption that $\rho > 0$ leads to the claimed lower bound on $|S'|$:

$$|S'| \geq \sigma m. \tag{6}$$

Since $\sigma < 1$, the bound $|S'| \geq \sigma m$ holds for the case $\rho = 0$ as well.

For each $i \in S'$, choose a set $P_i \subset \binom{\mathcal{D}}{2}$ of $|P_i| \geq \gamma\ell/2$ disjoint pairs of codewords in $\mathcal{D}_i$ such that for each $\{c_1, c_2\} \in \mathcal{P}_i$, we have $c_1[i] = c_2[i]$. This is always possible since $|A_i| \leq \ell$ and $|\mathcal{D}_i| \geq (1+\rho)^2|\mathcal{D}| \geq (1+\gamma)\ell$. Now choose $\mathcal{C}'$ randomly by including each element of $\mathcal{D}$ with probability $p = (\gamma\ell/2)^{-1/2}\ell(1+\alpha)|\mathcal{D}|^{-1}$. Since $\ell \geq 4\gamma^{-1}$ by hypothesis and $|\mathcal{D}| \geq \ell(1+\alpha)$ by the assumption that $\mathcal{C}_S$ is not $(\rho, \ell, \ell(1+\alpha))$-list recoverable, we have $p < 1$. The expected size of $\mathcal{C}'$ is

$$\mathbf{E}[|\mathcal{C}'|] = p|\mathcal{D}| \leq (\gamma/(2\ell))^{-1/2}(1+\alpha) \leq (8\ell/\gamma)^{1/2}.$$

We remark that this is the only place where we use the assumption that $\alpha < 1$. For any fixed pair $c_1 \neq c_2$ of codewords in $\mathcal{D}$, the probability that both are included in $\mathcal{C}'$ is p^2. Since the pairs in P_i are disjoint, the events that two distinct pairs $\{c_1, c_2\}, \{c_3, c_4\} \in P_i$ are both included in $\mathcal{C}'$ are independent. Hence, the probability that no pair in P_i is included in $\mathcal{C}'$ is $(1-p^2)^{|P_i|} < e^{-p^2|P_i|} < 1/2$. Consequently, for each fixed $i \in S'$, the probability that $i \in T(\mathcal{C}')$ is greater than $1/2$. By linearity of expectation, $\mathbf{E}[|T(\mathcal{C}') \cap S'|] \geq |S'|/2 \geq \sigma m/2$.

Let

$$Y = |T(\mathcal{C}') \cap S'| - \frac{\sigma m}{4}\frac{|\mathcal{C}'|}{\mathbf{E}[|\mathcal{C}'|]}.$$

By linearity of expectation, $\mathbf{E}[Y] \geq \sigma m/4$, hence there is some specific choice of $\mathcal{C}'$ for which $Y \geq \sigma m/4$. This can hold only if $|T(\mathcal{C}') \cap S| \geq |T(\mathcal{C}') \cap S'| \geq m/4$ and $|\mathcal{C}'| \leq 3\mathbf{E}(|\mathcal{C}'|)$ simultaneously, which establishes (1) and (2).

Next we bound the probability that, for a fixed choice of $\mathcal{C}'$ and random S, we have have $|T(C') \cap S|$ large. Let $\mathcal{C}' \subset \mathcal{C}$ be an arbitrary set of $|\mathcal{C}'| \leq 10\ell^{1/2}\gamma^{-1/2}$ codewords. Since the distance of $\mathcal{C}'$ is at least $n - nq^{-1} - d$ and $d \geq nq^{-1}$, we have

$$|T(\mathcal{C}')| \leq (nq^{-1} + d)\binom{|\mathcal{C}'|}{2} < d|\mathcal{C}'|^2. \tag{7}$$

For $S \in \binom{[n]}{m}$ chosen uniformly at random, $|T(\mathcal{C}') \cap S|$ follows a hypergeometric distribution. Specifically, we are making m draws from a population size of n of which $|T(\mathcal{C}')| \leq d|\mathcal{C}'|^2$ contribute to $|T(\mathcal{C}') \cap S|$. Using the assumption that $\ell \leq \gamma\sigma n(800d)^{-1}$, the expected value of $|T(\mathcal{C}') \cap S|$ is

$$\mathbf{E}\left[|T(\mathcal{C}') \cap S|\right] \leq d|\mathcal{C}'|^2 n^{-1} m \leq 100\frac{d\ell}{\gamma n}m \leq \frac{\sigma m}{8}. \tag{8}$$

Next we use the following large deviation inequality for hypergeometric random variables (see [3]). Let X be a hypergeometric random variable with mean μ. Then for any $\alpha \geq 1$,

$$\mathbb{P}(X \geq (1+\alpha)\mu) \leq \exp(-\alpha\mu/4). \tag{9}$$

Together with (8), this gives

$$\mathbb{P}(|T(\mathcal{C}') \cap S| \geq \sigma m/4) \leq \exp\left(-\frac{\sigma m}{32}\right). \tag{10}$$

Finally, we take a union over all candidates for $\mathcal{C}'$. Let X be the event that $\mathcal{C}_S$ is not (ℓ, α, ρ) list recoverable, with $S \in \binom{[n]}{m}$ uniformly at random. Using the assumption that $\sigma m \geq 1280\sqrt{\ell/\gamma}\log|\mathcal{C}|$, we have

$$\begin{aligned}
\mathbb{P}(X) &\leq \sum_{\mathcal{C}' \subset \mathcal{C}_S : |\mathcal{C}'| \leq 10\sqrt{\ell/\gamma}} \mathbb{P}(|T(\mathcal{C}' \cap S)| \geq \sigma m/4) \\
&\leq \binom{|\mathcal{C}|}{\lceil 10\sqrt{\ell/\gamma}\rceil + 1} \exp\left(-\frac{m}{32}\right) \\
&< \exp\left(20\sqrt{\ell/\gamma}\log|\mathcal{C}| - \sigma m/32\right) \\
&\leq \exp(-\sigma m/64),
\end{aligned}$$

as claimed. ◀

We now show how to derive Theorem 2 from Theorem 10.

Proof of Theorem 2. Suppose we have α, ρ, n, q, and ϵ as in the hypotheses of Theorem 2. Let $\gamma = (1+\alpha)(1-\rho)^2 - 1$, $\sigma = (1-\rho)(2-\rho)^{-1}$ and $m = \lceil 1280\epsilon^{-1}\log|C|\rceil$. The singleton bound combined with the assumption that $\epsilon < c$ for a suitably chosen absolute constant c implies that $m < n$. Choose $S \in \binom{[n]}{m}$ uniformly at random. The rate of $\mathcal{C}_S$ is

$$R = \log|\mathcal{C}|(m\log q)^{-1} = \Omega(\epsilon(\log q)^{-1}).$$

It is straightforward to check that the hypotheses of Theorem 10 are satisfied if we take $\ell = \epsilon^{-2}\sigma^2\gamma$, and hence we have that $\mathcal{C}_S$ is $(\rho, \ell, \ell(1+\alpha))$-list recoverable with high probability. ◀

4 Upper bound

Here we show the aforementioned upper bound for the rate to which a degree-d Reed-Solomon code over $\mathbb{F}_q$ can be randomly punctured to be $(Omega(q), 1/2)$-zero-error list-recoverable.

First, we recall a bit of standard and relevant sumset notation. For a group G and subsets $A, B \subseteq G$, we denote the sumset $A + B = \{a + b \mid a \in A,\ b \in B\}$. Clearly, we have $|A + B| \leq |A| \cdot |B|$. If $G = \mathbb{Z}_p$, then for $n < p/2$, we have that $[n] + [n] = \{2, \ldots, 2n\}$. We are now ready to state and prove the upper bound.

▶ **Theorem 11.** *Let $m = o(\log q)$, and S be a uniformly random subset of $\mathbb{F}_q$ of size m where q is a large prime. Then for every $d \geq 1$, the degree-d Reed-Solomon code with the evaluation set at S is, with high probability, not $(\Omega(q), 1/2)$-zero-error list-recoverable.*

Proof. Let $S = \{s_0, \ldots, s_m\}$. Let t be a large number such that $t^m = o(\sqrt{q})$. We are using the fact that $m = o(\log q)$ for the existence of such a t. W.L.O.G assume $s_0 = 0$ and $s_1 = 1$ (if $0, 1 \notin S$, then adding them to S only makes the lower bound stronger). Consider the two sets

$$A_0 = \frac{1}{1 - s_2}[t] + \cdots \frac{1}{1 - s_{m-1}}[t]$$

and

$$A_1 = \frac{1}{s_2}[t] + \cdots \frac{1}{s_{m-1}}[t].$$

▷ Claim 12. With high probability over the choice of S, we have that $|A_0|, |A_1| = \Omega(\left(t^{m-2}\right))$.

Proof. We do the proof for A_0, the case for A_1 follows analogously. Let P be the set of "collisions" in A_0. Formally:

$$P := \left\{ (a_2, \ldots, a_{m-2}, b_2, \ldots, b_{m-2}) \mid \sum_{i=2}^{m-2} a_i s_i = \sum_{i=2}^{m-2} b_i s_i \right\}.$$

So the number of distinct elements in A_0 is at least $t^{m-2} - |P|$. We observe that

$$\begin{aligned} \mathbf{E}[|P|] &= \sum_{\substack{a_2,\ldots,a_{m-2} \in [t] \\ b_2,\ldots,b_{m-2} \in [t]}} \mathbb{P}\left(\sum_{i=2}^{m-2} a_i s_i = \sum_{i=2}^{m-2} b_i s_i \right) \\ &\leq \frac{1}{p} t^{2m-4} \\ &= o(t^{m-2}). \end{aligned}$$

So by Markov's Inequality, with high probability, $|A_0| \sim t^{m-2}$. ◀

Consider $\mathcal{D}$, the set of degree-1 Reed-Solomon codes given by the lines

$$\{Y = aX + b\}_{b \in A_0, a \in A_1}.$$

First, we note that $|Y| = \Omega(t^{2m-4})$. Geometrically, $\mathcal{D}$ is just the set of all lines passing through some point of $\{0\} \times A_0$ and $\{1\} \times A_1$. Clearly, $\{c[0] \mid c \in \mathcal{C}\} = A_0$ and $\{c[1] \mid c \in \mathcal{D}\} = A_1$. For $i \neq 0, 1$, let us similarly define $A_i := \{c[s_i] \mid c \in \mathcal{D}\}$.We have that

$$\begin{aligned}
A_i &= \{a(1-s_i) + bs_i\}_{b\in A_0, a\in A_1} \\
&= (1-s_i)\left(\frac{1}{1-s_2}[t] + \cdots \frac{1}{1-s_{m-1}}[t]\right) + s_i\left(\frac{1}{s_2}[t] + \cdots \frac{1}{s_{m-1}}[t]\right) \\
&= \left([t] + \sum_{2\le j\le m,\ j\neq i} \frac{1-s_i}{1-s_j}[t]\right) + \left([t] + \sum_{2\le j\le m,\ j\neq i} \frac{s_i}{s_j}[t]\right) \\
&= \{2, \ldots, 2t\} + \sum_{2\le j\le m,\ j\neq i} \frac{1-s_i}{1-s_j}[t] + \sum_{2\le j\le m,\ j\neq i} \frac{s_i}{s_j}[t].
\end{aligned}$$

Thus, $|A_i| \le (2t) \times t^{2m-6} \le 2t^{2m-5}$.

This shows that there are lists $A_0, A_1, \ldots, A_m$ each of size at most $\ell := 2t^{2m-5}$ such that there are at least $\Omega(t^{2m-4}) = \ell^{1+\frac{1}{2m}}$ codewords, namely $\mathcal{D}$, contained in $A_0 \times \cdots \times A_m$. ◀

For a fixed d, the above theorem rules out hope of randomly puncturing degree-d Reed-Solomon codes to rate $\omega\left(\frac{1}{\log q}\right)$ for the desired list recoverability. We believe that this is essentially the barrier. We state the concrete conjecture that we alluded to in Section 2.2.

▶ **Conjecture 13.** *The degree-d Reed-Solomon code with evaluation set $\mathbb{F}_q$ can be randomly punctured to rate $\Omega_d\left(\frac{1}{\log q}\right)$ so that is it $(\Omega(q), 1/2)$-list recoverable with high probability.*

5 Discussion and open problems

The main open problem that we would like to showcase is Conjecture 13. This was probably believed to be true but we could not find it written down explicitly in the literature. List recoverable codes have connections to various other combinatorial objects (see [12]) and if true, Conjecture 13 could lead to the construction of some other interesting combinatorial objects.

The second open problem is to derandomize Theorem 2, i.e., to find an *explicit* Reed-Solomon code which is list recoverable beyond the Johnson bound at least in the zero-error case. Understanding how these evaluation sets look like could lead to progress on Conjecture 13, or could be interesting in its own right.

Finally, the last open problem is that given a Reed-Solomon code $\mathcal{C} \subset [q]^m$ of rate R on a randomly chosen evaluation set S, find an efficient algorithm for list recovery, i.e., take input lists $A_1, \ldots, A_m$ of size $O(R^{-2}(\log q)^{-1})$, and output all the codewords contained in $A_1 \times \cdots \times A_m$ with high probability (over the choice of S and the randomness used by the algorithm). This would also likely require some understanding of the properties of the evaluation set.

References

1 Xue Chen. personal communication.

2 Xue Chen and David Zuckerman. Existence of simple extractors. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:116, 2018. URL: `https://eccc.weizmann.ac.il/report/2018/116`.

3 Devdatt Dubhashi and Alessandro Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2009.

4 Sivakanth Gopi, Swastik Kopparty, Rafael Mendes de Oliveira, Noga Ron-Zewi, and Shubhangi Saraf. Locally testable and locally correctable codes approaching the Gilbert-Varshamov bound. *IEEE Trans. Information Theory*, 64(8):5813–5831, 2018. doi:10.1109/TIT.2018.2809788.

5 Venkatesan Guruswami. personal communication.

6 Venkatesan Guruswami. Algorithmic results in list decoding. *Foundations and Trends in Theoretical Computer Science*, 2(2), 2006. doi:10.1561/0400000007.

7 Venkatesan Guruswami and Piotr Indyk. Expander-based constructions of efficiently decodable codes. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 658–667. IEEE, 2001.

8 Venkatesan Guruswami and Atri Rudra. Limits to list decoding Reed-Solomon codes. *IEEE Trans. Information Theory*, 52(8):3642–3649, 2006. doi:10.1109/TIT.2006.878164.

9 Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh-vardy codes. *J. ACM*, 56(4):20:1–20:34, 2009. doi:10.1145/1538902.1538904.

10 Atri Rudra and Mary Wootters. Every list-decodable code for high noise has abundant near-optimal rate puncturings. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 764–773, 2014. doi:10.1145/2591796.2591797.

11 Atri Rudra and Mary Wootters. It'll probably work out: Improved list-decoding through random operations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 287–296, 2015. doi:10.1145/2688073.2688092.

12 Salil P. Vadhan. The unified theory of pseudorandomness: guest column. *SIGACT News*, 38(3):39–54, 2007. doi:10.1145/1324215.1324225.

On Perturbation Resilience of Non-Uniform k-Center

Sayan Bandyapadhyay
Department of Informatics, University of Bergen, Norway
sayan.bandyapadhyay@gmail.com

Abstract

The Non-Uniform k-center (NUkC) problem has recently been formulated by Chakrabarty, Goyal and Krishnaswamy [ICALP, 2016] as a generalization of the classical k-center clustering problem. In NUkC, given a set of n points P in a metric space and non-negative numbers $r_1, r_2, \ldots, r_k$, the goal is to find the minimum dilation α and to choose k balls centered at the points of P with radius $\alpha \cdot r_i$ for $1 \leq i \leq k$, such that all points of P are contained in the union of the chosen balls. They showed that the problem is NP-hard to approximate within any factor even in tree metrics. On the other hand, they designed a "bi-criteria" constant approximation algorithm that uses a constant times k balls. Surprisingly, no true approximation is known even in the special case when the r_i's belong to a fixed set of size 3. In this paper, we study the NUkC problem under perturbation resilience, which was introduced by Bilu and Linial [Combinatorics, Probability and Computing, 2012]. We show that the problem under 2-perturbation resilience is polynomial time solvable when the r_i's belong to a constant sized set. However, we show that perturbation resilience does not help in the general case. In particular, our findings imply that even with perturbation resilience one cannot hope to find any "good" approximation for the problem.

2012 ACM Subject Classification Theory of computation → Computational geometry; Mathematics of computing → Approximation algorithms

Keywords and phrases Non-Uniform k-center, stability, clustering, perturbation resilience

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.31

Category APPROX

Related Version A full version of the paper is available at `https://arxiv.org/abs/2004.12633`.

Funding This work is partly supported by the Research Council of Norway via the project "MULTIVAL".

Acknowledgements I am indebted to Kasturi Varadarajan and Tanmay Inamdar for several helpful discussions and for giving invaluable feedback on this work. I also thank the anonymous reviewers whose suggestions have helped to further improve the quality of the paper.

1 Introduction

Stability is a popular notion, which has been used in literature in the context of *beyond worst case analysis*. The general idea is to impose extra constraints on the inputs such that the (stable) instances that satisfy those constraints can capture the instances that appear in real life applications. In other words, we would like to exclude the "unrealistic" instances from consideration and obtain optimistic bounds for algorithms on the remaining inputs. For example, a major collection of work along this line have focused on designing polynomial time algorithms for NP-complete problems under different stability conditions. Bilu and Linial [10] introduced a notion of stability, which they termed as *ψ-perturbation resilience* for some $\psi > 1$. Informally, an instance is called ψ-perturbation-resilient if the optimal solution remains same even after the instance is perturbed by a factor of ψ.


© Sayan Bandyapadhyay;
licensed under Creative Commons License CC-BY

Recently, researchers have shown sufficient interest in studying geometric clustering problems under perturbation resilience. An instance of a clustering problem is ψ-perturbation-resilient if the optimal clustering is unique and remains unchanged under ψ-factor perturbation of the input distances. Awasthi et al. [6] showed that the standard center based clustering problems (e.g. k-center, k-median) can be solved in polynomial time under ψ-perturbation-resilience for $\psi \geq 3$. In any such center based clustering problem, the clustering is obtained by assigning a point to its nearest center. In other words, the clustering is induced by the Voronoi partition of the points w.r.t. the chosen centers. Subsequently, Balcan and Liang [8] designed a polynomial time algorithm for these clustering problems under ψ-perturbation-resilience for $\psi \geq 1 + \sqrt{2}$, improving the bound of Awasthi et al. [6]. Later, Balcan et al. [7] improved the bound for k-center to 2. On the other hand, they showed that k-center under ψ-perturbation-resilience cannot be solved in polynomial time for $\psi < 2$, unless $\mathsf{NP} = \mathsf{RP}$. They also considered the more general asymmetric k-center problem, where the distances are not necessarily symmetric (but satisfy triangle inequality). The problem is known to not admit a constant approximation unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{\log \log n})$, where n is the input size [15]. Surprisingly, Balcan et al. [7] showed that asymmetric k-center under 2-perturbation-resilience can be solved in polynomial time. Angelidakis et al. [4] gave a generic polynomial time algorithm for clustering problems with center based objectives (e.g. k-center, k-median, k-means) under 2-perturbation-resilience. Recently, Cohen-Addad and Schwiegelshohn [16] proved that a simple local search scheme yields optimal solutions for problems like k-median and k-means, under ψ-perturbation-resilience for $\psi > 3$. Chekuri and Gupta [14] showed that an LP relaxation of k-center under 2-perturbation-resilience admits an integral solution. They also proved the same result for k-center with outliers. Balcan and Liang [8] introduced a weaker stability assumption called (ψ, ϵ)-perturbation-resilience, where the optimal solution under ψ-perturbation can differ in at most ϵ fraction of the points from the original optimal clustering (see Preliminaries for the formal definition). Assuming that each cluster contains more than $2\epsilon n$ points, Balcan et al. [7] showed that k-center under $(3, \epsilon)$-perturbation-resilience can be solved in polynomial time, where n is the number of input points.

The increasing interest in studying perturbation resilient clustering has given rise to several open directions. One such interesting direction is to study clustering problems, where the clustering is not necessarily induced by Voronoi partition. One such clustering problem is *Non-Uniform k-center* (NUkC). In NUkC, we are given a set of n points P in a metric space, non-negative integers $r_1, r_2, \ldots, r_k$, and the goal is to find the minimum dilation α and to choose k balls centered at the points of P with radius $\alpha \cdot r_i$ for $1 \leq i \leq k$, such that all points of P are contained in the union of the chosen balls. We refer to any feasible solution of this problem composed of the chosen balls as a *feasible placement*. From a feasible placement, a clustering is retrieved in the following way – each point is assigned to a fixed ball that contains the point, and then for each ball, the points that are assigned to that ball form a cluster. Figure 1 shows that, the optimal clustering for an instance of NUkC is not the same as the Voronoi partition w.r.t. the centers of the balls in the optimal placement. The NUkC problem was formulated by Chakrabarty et al. [13] as a generalization of the well-studied k-center clustering problem, where all r_i's are same. Apart from clustering, NUkC has several applications in vehicle routing, sensor placement, and so on. For example, in vehicle routing, we need to find k depot locations corresponding to k vehicles having different speeds, such that any customer can be served by some vehicle as quickly as possible.

As mentioned before, k-center is a special case of NUkC where all the input radii are equal. We call this version of the problem as NUkC with one radius class. In general, all the radii might not be equal. But, we can consider only distinct radii from the input and

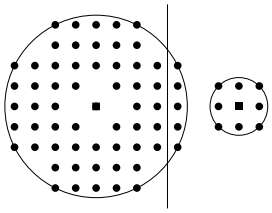


Figure 1 The optimal clusters are contained in the two disks in the Euclidean plane. The centers of balls are shown by boxes. The Voronoi partition w.r.t. the centers contains two subsets of points lying on the different sides of the vertical bisector line.

associate a multiplicity parameter k_i, with each such radius r_i, which denotes the number of balls of radius r_i that can be opened. Then the problem can be formulated equivalently in the following way.

▶ **Definition 1** (NUkC with t radii classes)**.** *Given a set of n points P in a metric space, $t \leq k$ distinct radii $r_1 > r_2 > \ldots > r_t$ and non-negative integers $k_1, \ldots, k_t$ such that $\sum_{i=1}^{t} k_i = k$, the goal is to find the minimum dilation α and to choose k_i balls centered at the points of P with radius $\alpha \cdot r_i$ for all $1 \leq i \leq t$, such that the union of the chosen balls contains all the input points.*

We note that k-center with outliers is a special case of NUkC with 2 radii classes where the radius $r_2 = 0$. Using a reduction from the Firefighters problem [1], Chakrabarty et al. (Theorem 2 in [13]) proved that NUkC is NP-hard to approximate within any constant factor even in tree metrics. In fact, their construction proves c-inapproximability of the problem for any c, not necessarily a constant. On the other hand, they designed a (c_1, c_2) bi-criteria approximation for the problem for large constants c_1 and c_2, i.e., if the algorithm is allowed to use $c_1 \cdot k_i$ balls of type i (thus $c_1 \cdot k$ in total), it can produce a solution with dilation at most c_2 times the optimal dilation. They also gave a $1 + \sqrt{5}$-approximation for NUkC with two radii classes. For k-center with outliers, they gave an improved 2-approximation. However, even when the number of distinct radii is 3, no true approximation is known.

The motivation behind the study of NUkC under perturbation resilience is that, in many applications, the distance function is heuristic. In fact, when the points represent structures like images, proteins, documents, etc., it is very hard to find the true distance function, and various standard distance/dissimilarity measures are used. If one solves a clustering problem with such a heuristic distance function and expects good results, then they implicitly assume that the optimal solution of the problem is not sensitive to small perturbations of the distance function. The perturbation resilience condition is a natural way to make this implicit assumption precise. And, the separation between the clusters forces an optimal clustering to be unique.

Our results. In this paper, we obtain the following results.

1. *Polynomial time exact algorithm for NUkC with a constant number of radii classes under "2-perturbation-resilience" and "$(3, \epsilon)$-perturbation-resilience when each cluster contains more than $\epsilon n + 1$ points".* Our algorithm reduces the NUkC problem to a version of Firefighters problem on trees (formally defined in Section 4). Under the stability assumptions, we can show that a feasible solution of NUkC maps to a feasible solution of Firefighters problem and vice versa. Here, in particular, we use the "well-separated" structure of the clusters in the optimal clustering that follows due to stability. The

reduction has the property that if NUkC has t distinct radii classes, then the height of the constructed tree instance is $t+1$. Then we show that using a dynamic programming based scheme the Firefighters problem can be solved in polynomial time for constant height tree instances. Thus we also obtain a polynomial time algorithm for NUkC under perturbation resilience with a constant number of radii classes. We note that the algorithms for center based clustering problems in [4, 8, 14] are also based on tree computation and dynamic programming. However, the structure of the tree we compute is very different. We also note that our result under 2-perturbation-resilience is tight, as even for k-center it is unlikely to obtain a polynomial time algorithm under ψ-perturbation-resilience for $\psi < 2$. To prove the result for $(3, \epsilon)$-perturbation-resilience, we assume that each cluster contains more than $\epsilon n + 1$ points. We note that such a lower bound is necessary, as in its absence even k-center is NP-hard [7] under (ψ, ϵ)-perturbation-resilience for all $\psi \geq 1$ and $\epsilon > 0$.

2. *γ-inapproximability for NUkC under ψ-perturbation-resilience for any $\gamma > 1$ and $\psi \leq \gamma$, unless* NP = RP. Our result implies that, for any $\psi, \gamma > 1$, even with ψ-perturbation-resilience one cannot hope to find a γ-approximation for the problem. Our result should be contrasted with the polynomial time algorithm for asymmetric k-center under 2-perturbation-resilience, as asymmetric k-center is another candidate problem which is hard to approximate within a constant factor. To prove the result, we use a chain of reductions starting from the satisfiability problem to the NUkC problem in tree metrics under perturbation resilience assumption. The last reduction in the chain is from a version of the Firefighters problem which shows that NUkC is hard to approximate within a factor of γ in tree metrics for any γ. Our reduction is similar to the reduction in [13]. Then, we argue that the constructed tree instances of NUkC are γ-perturbation-resilient, and hence the similar hardness follows even for NUkC under γ-perturbation-resilience. We also extend this hardness result to Euclidean metric of dimension d for $d \geq 1$ using a classical tree embedding result of Gupta [21].

The main contribution of this paper is twofold. The first one is to be able to establish an exact connection between NUkC under perturbation resilience and the Firefighters problem on trees. To establish this connection, we need to prove that perturbation resilience implies that the optimal clusters are "well-separated". Similar properties have been proved in the context of other problems (e.g., k-center). Our contribution is to be able to extend these proofs for NUkC as well. However, the extension is non-trivial, and one need sufficiently good amount of extra work, as here we need to deal with non-uniform radii. We note that Chakrabarty et al. [13] also showed a reduction from NUkC to Firefighters. However, their LP-aware reduction is very different. Our second contribution is the tight hardness result for the problem. This result along with the polynomial time algorithm gives the complete picture for NUkC under perturbation resilience. To prove this result we are faced with the following challenges. In any such hardness construction, one needs to show that the instances of NUkC to which we map are perturbation resilient. Thus, we need to show that these instances have unique optimal solution and the optimal solution does not change with some perturbation of the distances. Chakrabarty et al. [13] showed a reduction from Firefighers to NUkC. However, using their distance function it is not straightforward to show that the constructed instances are insensitive to the perturbation of distances. Nevertheless, we consider a similar distance function and show the reduction works out well with this modification. To prove the uniqueness of the optimal solutions, we reduce a "unique" version of 3SAT to a "unique" version of Firefighters using a chain of reductions.

Related work and Open questions. Other optimization problems have also been studied under stability assumptions [4, 17, 19, 25, 26]. Also different stability assumptions have been introduced and well-studied in the literature [5, 24, 27]. Most of the clustering problems mentioned here are NP-hard, but admit some constant approximations, e.g., see [3, 12, 20, 22] and the references therein. It would be interesting to see if one can obtain a constant approximation for NUkC with a constant number of radii classes without any perturbation resilience assumptions. Also, one can study similar hard clustering problems (e.g., k-clustering [9]) under perturbation resilience.

Organization. In Section 2, we define some notations that we use throughout the paper, and make a few observations that will be useful later. In Section 3, we list some properties implied due to perturbation resilience of the input instances. Then in Section 4, we discuss the algorithm for NUkC with any constant number of classes and prove its correctness by using the properties proved in the previous section. Lastly, in Section 5, we prove the hardness results for the general problem.

2 Preliminaries

We denote an instance of NUkC with t radii classes on metric d by (P, d, t). Note that the radii (r_i) and multiplicity (k_j) parameters remain implicit in this notation. But, references to these parameters will become clear from the context. A ball with center $p \in P$ and radius r, denoted by $B(p, r)$, is the set of points $\{q \in P \mid d(p, q) \leq r\}$. A set of balls *covers* a set of points if the union of the balls contains all the points. Recall that a feasible placement is a feasible solution of the problem composed of the chosen balls that cover all the input points. A feasible NUkC *clustering* $\mathcal{C}$ of the input set of points P is a partition $\{C_1, \ldots, C_k\}$, such that there is a feasible placement Π with the property that for all i, C_i is a subset of a ball in the placement. We say that the clustering $\mathcal{C}$ is *induced* by the placement Π. The radius of a cluster C w.r.t. any distance function d, denoted by c-radius(C, d), is $\min_{p \in P} \max_{q \in C} d(p, q)$. Note that no ball centered at a point $p \in P$ of radius smaller than c-radius(C, d) can cover all the points of C. For a placement with dilation α, a ball with radius αr_i (resp. $< \alpha r_i$ and $\geq \alpha r_i$) is called an r_i (resp. $< r_i$ and $\geq r_i$) -ball.

Consider a metric space P with metric $d : P \times P \to \mathbb{R}_{\geq 0}$. A metric d_1 is called a ψ-perturbation of d if for any $p, q \in P$, $d(p, q)/\psi \leq d_1(p, q) \leq d(p, q)$[1]. In this paper, all perturbations we consider satisfy the metric properties.

▶ **Definition 2.** *An instance $\mathcal{I} = (P, d, t)$ of NUkC is called ψ-perturbation-resilient (ψ-PR) if for any metric ψ-perturbation d_1 of d, the unique optimal NUkC clustering of $\mathcal{I}' = (P, d_1, t)$ is identical to the unique optimal clustering of $\mathcal{I}$.*

Note that in general, optimal clustering of NUkC might not be unique. We refer to the instance $\mathcal{I}'$ as a ψ-perturbed instance of $\mathcal{I}$. A few examples demonstrating the definition of perturbation resilience w.r.t. NUkC with t radii classes are shown in Appendix A. We also consider another notion of perturbation resilience introduced by Balcan and Liang [8], where the optimal clustering is allowed to be different by a few points when the distances are perturbed. Here we rewrite this notion in terms of NUkC. Two clusterings $\mathcal{C} = \{C_1, \ldots, C_k\}$ and $\mathcal{C}' = \{C'_1, \ldots, C'_k\}$ are called ϵ-close if at most ϵn points are clustered differently in the two clusterings, i.e., the minimum value of $\sum_{i=1}^{k} |C_i \setminus C'_{f(i)}|$ over all permutations f of $\{1, 2, \ldots, k\}$ is at most ϵn.

[1] One can also define ψ-perturbation by both increasing and decreasing the distances - the two definitions are equivalent modulo some factor, as one can always scale the input distances appropriately.

▶ **Definition 3.** *An instance $\mathcal{I} = (P, d, t)$ of NUkC is called (ψ, ϵ)-perturbation-resilient ((ψ, ϵ)-PR) if for any metric ψ-perturbation d_1 of d, any optimal NUkC clustering of $\mathcal{I}' = (P, d_1, t)$ is ϵ-close to any optimal clustering of $\mathcal{I}$.*

This is again a well-studied stability criterion [2]. Note that when $\epsilon = 0$, any optimal NUkC clustering of $\mathcal{I}'$ must be same as any optimal clustering of $\mathcal{I}$. This implies that optimal clustering of $\mathcal{I}$ and $\mathcal{I}'$ are unique and we obtain the definition of ψ-PR. Thus, if an instance of NUkC is ψ-PR, then it is also $(\psi, 0)$-PR, and hence any hardness result for NUkC under ψ-PR trivially follows for NUkC under (ψ, ϵ)-PR. Now, we have the following simple observation, which will be useful later in proving the properties of the PR instances.

▶ **Observation 4.** *Consider an NUkC instance $\mathcal{I} = (P, d, t)$ that admits a unique optimal clustering $\mathcal{O}$. Let C be any cluster in $\mathcal{O}$. Also, consider an optimal placement Π where C is covered by a ball B. Then, the following two properties hold.*

- *The center p of the ball B must belong to C.*
- *For any two points u, v that lie in two different clusters of $\mathcal{O}$, both of u, v cannot be contained in B.*

Proof.

- Suppose p belongs to the cluster C' such that $C \neq C'$. Construct another clustering $\mathcal{O}'$ by selecting all the clusters in $\mathcal{O}$ except C and C', and the clusters $C \cup \{p\}$ and $C' \setminus \{p\}$. It is not hard to see that $\mathcal{O}'$ is also a feasible clustering induced by Π. As Π is an optimal placement, $\mathcal{O}'$ is also an optimal clustering, which contradicts the uniqueness of the optimal clustering of $\mathcal{I}$. Hence, the statement follows.
- Suppose B contains both u and v. We construct a new clustering $\mathcal{O}'$, which is identical to $\mathcal{O}$ except, in $\mathcal{O}'$, we move the points u, v to the cluster C. Note that the clustering $\mathcal{O}'$ can be induced by the placement Π, as the ball B that covers $C \in \mathcal{O}$ also contains u, v. Hence, $\mathcal{O}'$ is an optimal clustering for $\mathcal{I}$ different than $\mathcal{O}$, which is a contradiction, and thus the statement follows. ◀

WLOG we can assume that the optimal dilation of a ψ-PR or a (ψ, ϵ)-PR instance of NUkC is 1. Like in the general case without perturbation resilience, in this case also the assumption can be introduced by scaling r_i values by a guessed value of the optimal dilation α.

▶ **Lemma 5.** *Suppose there is a polynomial time algorithm $\mathcal{A}$ for the NUkC problem with t radii classes under ψ-PR (resp. (ψ, ϵ)-PR) with the properties that (i) for an instance which admits a feasible placement of balls with dilation 1, $\mathcal{A}$ returns "yes" and a feasible clustering, and (ii) for an instance which does not admit a feasible placement of balls with dilation 1, $\mathcal{A}$ returns "no". Then, the NUkC problem with t radii classes under ψ-PR (resp. (ψ, ϵ)-PR) can be solved in polynomial time.*

Proof. Consider any instance $\mathcal{I} = (P, d, t)$ of the NUkC problem with t radii classes under ψ-PR (resp. (ψ, ϵ)-PR). Let α be the optimal dilation. Note that we do not know the value of α. However, as the input metric is finite, there are only polynomial number of guesses for α. We use the following procedure to obtain the optimal clustering for $\mathcal{I}$. In each step, we guess a value α' for the optimal dilation in the increasing order of the values. We construct a new instance $\mathcal{I}'$ from $\mathcal{I}$ by only changing the radius r_i to $\alpha' \cdot r_i$ for all i. Then, we apply the algorithm $\mathcal{A}$ on the constructed instance. If $\mathcal{A}$ returns "no", we repeat the process with a different guess. Otherwise, the procedure terminates. We return the same clustering returned by $\mathcal{A}$ as the solution for the instance $\mathcal{I}$.

Now, we argue about the correctness of the procedure. First, we claim that $\mathcal{I}'$ is a ψ-PR (resp. (ψ, ϵ)-PR) instance. Before proving this claim we discuss its consequences. Note that if there is no feasible solution for $\mathcal{I}$ with dilation α', then with k_i balls of radius $\alpha' \cdot r_i$ for all i it is not possible to cover the input points. Hence, in this case, for the constructed instance, there is no feasible solution with dilation 1. Thus, the algorithm correctly returns "no" assuming $\mathcal{I}'$ is a ψ-PR (resp. (ψ, ϵ)-PR) instance. If there is a feasible solution for $\mathcal{I}$ with dilation α', then with k_i balls of radius $\alpha' \cdot r_i$ for all i one can cover the input points. Thus, in that case, for the constructed instance, there is a feasible solution with dilation 1. Hence, $\mathcal{A}$ correctly returns "yes" assuming $\mathcal{I}'$ is ψ-PR (resp. (ψ, ϵ)-PR). Thus, when $\alpha' = \alpha$, $\mathcal{A}$ returns "yes" and the returned clustering is optimal for $\mathcal{I}$. Now, we prove the claim.

▷ Claim 6. $\mathcal{I}'$ is a ψ-PR (resp. (ψ, ϵ)-PR) instance.

Proof. First, we show that the optimal clustering of $\mathcal{I}'$ is unique. Note that the optimal dilation of $\mathcal{I}'$ is α/α'. Suppose optimal clustering of $\mathcal{I}'$ is not unique. Then, there are two different clusterings where the points can be covered using k_i balls of radius $(\alpha/\alpha') \cdot \alpha' \cdot r_i = \alpha \cdot r_i$ from each class i. It follows that there are two different optimal clusterings for $\mathcal{I}$. But, this is a contradiction, and thus the optimal clustering of $\mathcal{I}'$ is unique. Note that the optimal clusterings of $\mathcal{I}$ and $\mathcal{I}'$ are identical. Let $\mathcal{C}$ be that clustering. Now, consider any ψ-perturbation d_1 of the input metric d and the ψ perturbed instance $\mathcal{I}'_1$ of $\mathcal{I}'$. Let $\mathcal{I}_1 = (P, d_1, t)$ be the corresponding ψ perturbed instance of $\mathcal{I}$. Also, let $\mathcal{C}'_1$ be the optimal clustering of $\mathcal{I}'_1$ with dilation α'_1. For the sake of contradiction, suppose $\mathcal{C}'_1$ is not identical (resp. ϵ-close) to $\mathcal{C}$. We argue that $\mathcal{C}'_1$ is also an optimal clustering of $\mathcal{I}_1$. But, this is a contradiction, as $\mathcal{I}_1$ is a ψ perturbed instance of $\mathcal{I}$ and $\mathcal{I}$ is a ψ-PR (resp. (ψ, ϵ)-PR) instance. Now, note that a placement that induces the clustering $\mathcal{C}'_1$ of $\mathcal{I}'_1$ uses k_i balls of radius $\alpha'_1 \cdot \alpha' \cdot r_i$ from each class i. Thus, $\mathcal{C}'_1$ is a clustering for $\mathcal{I}_1$ with dilation $\alpha'_1 \cdot \alpha'$. It is sufficient to argue that this dilation is optimal for $\mathcal{I}_1$. Suppose the optimal dilation is $< \alpha'_1 \cdot \alpha'$. Then, using k_i balls of radius $< \alpha'_1 \cdot \alpha' \cdot r_i$ from each class i all the points can be covered. Hence, there is a clustering for $\mathcal{I}'_1$ with dilation $< \alpha'_1$, which is a contradiction, and hence the claim follows. ◁

Finally, as the number of guesses for α is a polynomial, the procedure terminates in polynomial time. ◀

3 Properties of Perturbation Resilience

In this section, we show that perturbation resilience imposes useful structure on the optimal solution. First, we consider the instances under (ψ, ϵ)-perturbation resilience with $\psi = 3$ and prove an interesting property of the optimal clustering.

▶ **Lemma 7.** *Consider any optimal placement* Π *for a* $(3, \epsilon)$*-PR NUkC instance* $\mathcal{I} = (P, d, t)$ *with optimal dilation 1 where the size of each optimal cluster is* $> \epsilon n + 1$*. Let* C_1 *and* C_2 *be two clusters induced by two balls of* Π *with radii* r_i *and* r_j*, respectively with* $r_i \geq r_j$*. Then, for any* $p \in C_1$ *and* $q \in C_2$*,* $d(p, q) > r_i$*.*

Proof. Let $\mathcal{O}$ be an optimal clustering of $\mathcal{I}$ that is induced by Π and contains C_1, C_2 as clusters. For the sake of contradiction, suppose there are two points $p \in C_1$ and $q \in C_2$ such that $d(p, q) \leq r_i$. Then, we show that there is a 3-perturbation d' of d such that an optimal clustering of $\mathcal{I}' = (P, d', t)$ is not ϵ-close to $\mathcal{O}$. But, this gives a contradiction to the assumption that $\mathcal{I}$ is a $(3, \epsilon)$-PR instance, and hence the lemma follows.

To construct the 3-perturbation d' of d, we at first construct another metric d_1. Later we will scale d_1 to construct d'. Let $B_1 = B(c_1, r_i)$ and $B_2 = B(c_2, r_j)$ be the balls in Π that induce C_1 and C_2, respectively. Then, for any $s \in C_2$, $d(p,s) \leq d(p,q) + d(q,s) \leq r_i + 2r_j \leq 3r_i$. Also, for any $w \in C_1$, $d(p,w) \leq 2r_i$. First, we construct a complete graph G with vertex set equal to P, and for any edge (u,v), its length is defined by the function l as follows.

$$l(u,v) = \begin{cases} 3r_i & \text{if } u = p, v \in (C_1 \cup C_2) \setminus \{c_1\} \text{ and } d(u,v) \geq r_i \\ 3 \cdot d(u,v) & \text{otherwise} \end{cases}$$

The distance d_1 is the shortest path metric on G. Note that, as mentioned before, for any $v \in (C_1 \cup C_2) \setminus \{c_1\}$, $d(p,v) \leq 3r_i$. Thus, it is not hard to see that, for any $u, v \in P$, $d(u,v) \leq d_1(u,v) \leq 3 \cdot d(u,v)$. Now, let us define the metric d'. For any two points u, v, $d'(u,v) = d_1(u,v)/3$. Hence, for any $u, v \in P$, $d(u,v)/3 \leq d'(u,v) \leq d(u,v)$. It follows that d' is a metric 3-perturbation of d, and thus the optimal clustering of $\mathcal{I}'$ is ϵ-close to $\mathcal{O}$.

Now, let $\mathcal{I}_1 = (P, d_1, t)$ and $\mathcal{O}_1$ be an optimal clustering of $\mathcal{I}_1$. As scaling does not change optimality of a clustering (for a formal proof see the proof of Lemma 5), $\mathcal{O}_1$ is also an optimal clustering of the instance $\mathcal{I}' = (P, d', t)$. Thus $\mathcal{O}_1$ is ϵ-close to $\mathcal{O}$. Next, we prove the following claim.

▷ Claim 8. The optimal dilation of $\mathcal{I}_1$ is 3.

Proof. As for any $u, v \in V$, $d_1(u,v) \leq 3 \cdot d(u,v)$, the optimal dilation of $\mathcal{I}_1$ is at most 3. We prove that this dilation is at least 3. Suppose the dilation is less than 3. Let Π' be any placement with dilation less than 3 that induces the optimal clustering $\mathcal{O}_1$ of $\mathcal{I}_1$. Then, we show that $\mathcal{O}_1$ is also a feasible clustering of $\mathcal{I}$ with dilation less than 1. But, this is a contradiction, and hence the claim follows. Next, given Π', we show the existence of a placement for $\mathcal{I}$ with dilation less than 1 that induces $\mathcal{O}_1$.

Consider any cluster $C' \in \mathcal{O}_1$, and suppose it gets covered by an r_l-ball $B = B(w, r)$ in Π'. Let x be any point in C'. Now, consider the distance d_1. Let π be any shortest path between w and x. We claim that π cannot contain the edge (p,v) for any $v \in (C_1 \cup C_2) \setminus \{c_1\}$ with $d(p,v) \geq r_i$. For the sake of contradiction, say π contains (p,v). Note that $d_1(p,v) = 3r_i$. As π contains (p,v), $d_1(w,p) \leq r - 3r_i$. Now, consider any point $u \in (C_1 \cup C_2) \setminus \{c_1\}$. If $d(p,u) \geq r_i$, $d_1(p,u) = 3r_i$. Otherwise, $d(p,u) < r_i$, and thus $d_1(p,u) = 3 \cdot d(p,u) < 3r_i$. Thus, $d_1(w,u) \leq d_1(w,p) + d_1(p,u) \leq r$. Hence, all the points of $(C_1 \cup C_2) \setminus \{c_1\}$ are in B. But, as C_1, C_2 contain more than $\epsilon n + 1$ points, it follows that there is an optimal clustering of $\mathcal{I}_1$ that is not ϵ-close to $\mathcal{O}$. Thus, we get a contradiction. Hence, π does not contain (p,v), and thus from the definition of the metric d_1, it follows that $d_1(w,x) = 3 \cdot d(w,x)$. Thus, a ball centered at w and having radius $r/3$ can cover the points of C' in $\mathcal{I}$. Now, note that $r < 3r_l$, and thus $r/3 < r_l$. Hence, it is sufficient to use an r_l-ball with less than 1 factor expansion to cover the points of C' in $\mathcal{I}$. In our new placement for $\mathcal{I}$, we use the r_l-ball $B(w, r/3)$ corresponding to each such cluster C'. Clearly, the dilation of the new placement is less than 1. ◁

Now, we show a clustering $\mathcal{O}_2$ of $\mathcal{I}_1$ that contains exactly k clusters, has dilation 3 and is not ϵ-close to $\mathcal{O}$. $\mathcal{O}_2$ contains all the clusters in $\mathcal{O}$ except C_1 and C_2, and the clusters $(C_1 \cup C_2) \setminus \{c_1\}, \{c_1\}$. Note that for any $s \in (C_1 \cup C_2) \setminus \{c_1\}$, $d(p,s) \leq 3r_i$. Thus, $(C_1 \cup C_2) \setminus \{c_1\}$ can be covered by a ball of radius $3r_i$. It follows that the dilation of $\mathcal{O}_2$ is at most 3 and hence it is an optimal clustering. Clearly, the two clusterings $\mathcal{O}$ and $\mathcal{O}_2$ differ in $> \epsilon n$ points, as $|C_1| > \epsilon n + 1$ and $|C_2| > \epsilon n + 1$. Now, for the same reason mentioned before, $\mathcal{O}_2$ is also an optimal clustering of the instance $\mathcal{I}' = (P, d', t)$. Hence, d' is the desired 3-perturbation. This completes the proof of the lemma. ◀

In the proof of the above lemma, one could have defined d' directly without going via d_1. However, for simplicity of exposition, we have followed this approach. Indeed, this approach shows that if one defines ψ-perturbation by increasing the (instead of decreasing) distances, the lemma still holds. A proof can directly use the 3-perturbation d_1 in that case.

Note that, as a 3-PR instance is also a $(3,0)$-PR instance, the above lemma trivially follows for 3-PR instances. In the following, we will show that the above mentioned property of the optimal clustering follows even for any 2-PR instance.

▶ **Lemma 9.** *Consider any optimal placement* Π *for a 2-PR NUkC instance* $\mathcal{I} = (P, d, t)$ *with optimal dilation 1. Let* C_1 *and* C_2 *be two clusters induced by two balls of* Π *with radius* r_i *and* r_j*, respectively, where* $r_i \geq r_j$*. Then, for any* $p \in C_1$ *and* $q \in C_2$*,* $d(p,q) > r_i$*.*

Proof. Let $\mathcal{O}$ be the optimal clustering induced by the placement Π. Also, let B_1 and B_2 be the balls that induce the clusters C_1 and C_2, respectively. For the sake of contradiction, suppose there exist two points $p \in C_1$, $q \in C_2$ such that $d(p,q) \leq r_i$. The idea is to show that there is a metric d_1 that is a 2-perturbation of d such that $\mathcal{I}' = (P, d_1, t)$ has different optimal clustering than $\mathcal{O}$. But, this is a contradiction, and thus the lemma follows.

Let c_t be the center of the ball B_t for $t \in \{1,2\}$. Then, $d(c_1,q) \leq d(c_1,p) + d(p,q) \leq 2r_i$. We define the distance function d_1 in the following way. First, we construct the complete graph with vertex set equal to P, and for any edge (u,v), its length is defined by the function l.

$$l(u,v) = \begin{cases} \min\{d(u,v), r_i\} & \text{if } u = c_1 \text{ and } v = q \\ d(u,v) & \text{otherwise} \end{cases}$$

We note that, for any u, v, $d(u,v)/2 \leq l(u,v) \leq d(u,v)$. The distance function d_1 is defined by the shortest path distance between any pair of vertices. It is not hard to verify the following observation.

▶ **Observation 10.** d_1 *is a metric* 2*-perturbation of* d*.*

Hence, the instance $\mathcal{I}' = (P, d_1, t)$ has the same optimal clustering $\mathcal{O}$. Next, we prove a claim that the optimal dilation of $\mathcal{I}'$ is also 1.

▷ Claim 11. The optimal dilation of $\mathcal{I}'$ is 1.

Proof. As for any $u, v \in V$, $d_1(u,v) \leq d(u,v)$, the optimal dilation of $\mathcal{I}'$ is at most 1. We prove that this dilation is at least 1. Suppose the dilation is less than 1. Let Π' be any placement with dilation less than 1 that induces the clustering $\mathcal{O}$ of $\mathcal{I}'$. Then, we show that there is a placement for $\mathcal{I}$ with dilation less than 1. But, this is a contradiction, and hence the claim follows. Consider any cluster $C \in \mathcal{O}$ that gets covered by an r_t-ball $B = B(w,r)$ in Π'. Let x be any point in C. Now, consider the distance d_1. Let π be any shortest path between w and x. We claim that π cannot contain the edge (c_1, q). For the sake of contradiction, say π contains (c_1, q). But, this implies $d_1(w,c_1) \leq d_1(w,x) \leq r$ and $d_1(w,q) \leq d_1(w,x) \leq r$. Thus, B contains both c_1 and q. Now, by the first property of Observation 4, c_1 belongs to C_1. Thus, by the second property of Observation 4, we obtain a contradiction, as $q \in C_2$. Hence, π does not contain (c_1, q). It follows that $d_1(w,x) \geq d(w,x)$. Thus, the radius of the ball needed to cover the points of C in $\mathcal{I}$ is at most r. Hence, it is sufficient to use an r_t-ball with at most $r/r_t < 1$ factor expansion to cover the points of C in $\mathcal{I}$. Now, we construct a placement for $\mathcal{I}$ by selecting the same balls to cover the clusters that are used in Π'. Clearly, the dilation of this placement is less than 1. ◁

Next, we show that there is a different clustering $\mathcal{O}'$ of $\mathcal{I}'$ with exactly k clusters that achieves the optimal dilation. This gives rise to a contradiction, and thus $d(p,q) > r_i$. Now, there are two cases. In the first case, q is the only point in C_2, and thus $C_2 \setminus \{q\}$ is empty. In this case, we pick a non-singleton cluster C from $\mathcal{O} \setminus \{C_1\}$ and choose a point $s \in C$. Such a cluster exists WLOG. Then, we define $\mathcal{O}'$ to be the set of clusters in $\mathcal{O}$ except C, C_1 and C_2, and the clusters $C_1 \cup \{q\}, \{s\}$ and $C \setminus \{s\}$. In the second case, q is not the only point in C_2, and thus $C_2 \setminus \{q\}$ is not empty. In this case, $\mathcal{O}'$ is defined to be the set of clusters in $\mathcal{O}$ except C_1 and C_2, and the clusters $C_1 \cup \{q\}, C_2 \setminus \{q\}$. It is not hard to see that $C_1 \cup \{q\}$ can be covered by the ball $B(c_1, r_i)$. Also, if $C_2 \setminus \{q\}$ is not empty, then $B(c_2, r_j)$ covers the points in $C_2 \setminus \{q\}$. Hence, in all the cases, it is trivial to verify that the dilation of the new clustering is 1. ◀

Note that, in the above proof, to show that $\mathcal{O}'$ has dilation 1, we argue that there is a placement with dilation 1. The balls in the placement might not be disjoint (both $B(c_1, r_i)$ and $B(c_2, r_j)$ cover q). But, for the sake of just showing the optimality of the clustering, it is sufficient to show the existence of such a placement.

4 NUkC with a Constant Number of Radii Classes

In this section, we show a polynomial time reduction from NUkC to the Constrained Resource Minimization for Fire Containment on Trees problem.

▶ **Definition 12** (Constrained Resource Minimization for Fire Containment on Trees (CRMFC-T))**.** *Given a rooted tree $T = (V, E)$ with height $t+1$, a set of forbidden nodes $F \subseteq V$, and integers $k_1, \ldots, k_t$, the goal is to decide if there is a collection of non-root nodes $U \subseteq (V \setminus F)$ such that (a) for every leaf-root path π, U contains at least one node from π, and (b) $|U \cap L_i| \leq k_i$ for $1 \leq i \leq t$, where L_i is the layer i nodes of T, i.e., the nodes at distance exactly i from the root.*

Given any instance $\mathcal{I} = (P, d, t)$ of NUkC under 2-PR or $(3, \epsilon)$-PR (the size of each optimal cluster is more than $\epsilon n + 1$), we will show how to construct an instance $\mathcal{I}'$ of CRMFC-T such that $\mathcal{I}$ has a feasible placement with dilation 1 iff $\mathcal{I}'$ has a feasible solution. Also, from a feasible solution for $\mathcal{I}'$, a feasible solution for $\mathcal{I}$ can be computed in polynomial time. In the constructed instance $\mathcal{I}'$, the height of the tree is one more than the number of radii classes in NUkC. We show that CRMFC-T can be solved in polynomial time if the height of the input tree is a constant. From Lemma 5, it follows that the perturbation resilient version of NUkC can be solved in polynomial time if the number of classes is a constant. Thus, we obtain the following theorem.

▶ **Theorem 13.** *NUkC under 2-PR (or $(3, \epsilon)$-PR, where the size of each optimal cluster is more than $\epsilon n + 1$) can be solved in polynomial time if the number of radii classes is a constant.*

4.1 Tree Construction

Let G be the complete graph that defines the distances between the input points. Note that we are also given the input radii $r_1 > r_2 > \ldots > r_t$. We construct the tree T in t rounds that contains t levels other than the root level. We denote the nodes at level i by L_i for $i \in \{0, \ldots, t\}$. L_0 contains a singleton node – the root of the tree. For $i \geq 1$, in i^{th} round,

we construct the nodes L_i and connect them with the nodes in L_{i-1}. Each node v in T corresponds to a connected subgraph G_v of G. The root corresponds to G itself. Also, each node is marked with either yes or no denoting if the node can be selected or it is in the forbidden set.

For each index $i \in \{1, \ldots, t\}$, in i^{th} round, we consider all the nodes $v \in L_{i-1}$ and the subgraph G_v corresponding to v. We remove all the edges with weight more than r_i from G_v. Let $G_v^1, \ldots, G_v^l$ be the connected components formed from G_v due to the removal of these edges. We add l children of v to L_i corresponding to these connected l subgraphs. For each such child u, if there is a node w in G_u, such that for all node x in G_u, $d(w, x) \leq r_i$, we label u with yes. Otherwise, we label u with no (forbidden). Lastly, for each level $i \geq 1$, the number of nodes that can be chosen from L_i in CRMFC-T is set to k_i. The following lemma establishes the connection between the two instances $\mathcal{I}$ and $\mathcal{I}'$.

▶ **Lemma 14.** *$\mathcal{I}$ has a feasible placement with dilation 1 iff $\mathcal{I}'$ has a feasible solution to CRMFC-T.*

Proof. First, suppose there is a feasible solution to $\mathcal{I}'$. For each chosen node v, v must be a yes node. Let i be the integer such that $v \in L_i$. Then, the points in G_v can be covered by an r_i ball centered at some point in G_v. We choose this ball in our placement. Note that we select at most k_i balls of radius r_i for all i. We prove that each point is covered in the constructed placement. Consider any point p. The way we construct the tree, each point can lie in the connected subgraph G_v of exactly one node v of L_j for all j. Let π be the root-leaf path in T, such that for any $v \in \pi$, p is in G_v. Now, there must be a node along π that is chosen in the solution of CRMFC-T. Let u be such a node. As we place a ball of radius r_i that covers all the points of G_u, p gets covered. Thus, $\mathcal{I}$ has a feasible placement with dilation 1.

Now, suppose $\mathcal{I}$ has a feasible placement with dilation 1. Let $\mathcal{O}$ be the clustering induced by the placement. Now, consider any cluster $C \in \mathcal{O}$, which is covered by a ball of radius r_j. Thus, c-radius$(C, d) \leq r_j$. The way the tree T is constructed it follows that all the points in C remain in the same connected subgraph G_v corresponding to a unique vertex $v \in L_i$ for each $i \leq j$. Let G_u be the subgraph corresponding to level $j-1$. As $\mathcal{I}$ is a 2-PR (resp. $(3, \epsilon)$-PR) instance, from Lemma 9 (resp. Lemma 7), we know that, for any $p \in C$ and $q \in P \setminus C$, $d(p, q) > r_j$. Thus, when the edges with weight more than r_j are removed from G_u, p and q cannot remain in the same component. But, as c-radius$(C, d) \leq r_j$ all the points of C remain in the same component. Also, by the first property of Observation 4, the center of the r_j-ball that covers C must lie in C. It follows that there is a yes node $C(v) \in L_j$ such that $G_{C(v)}$ contains only the points of C as vertices. For each cluster $C \in \mathcal{O}$, we select the yes node $C(v)$ in the solution to CRMFC-T. It is not hard to see that we choose at most k_j nodes from L_j. Now, consider any root-leaf path π in T corresponding to a leaf l. Let p be a point in G_l. Also, let p be a point in the cluster $C \in \mathcal{O}$. Then, there must be a yes node $C(v)$ in π such that $G_{C(v)}$ contains only the points of C. As we choose v in our solution, we have at least one node from the path π. Hence, the constructed solution is feasible. ◀

4.2 The Algorithm for CRMFC-T

In this section, we design a dynamic programming based algorithm that decides the feasibility of any instance of CRMFC-T. The algorithm runs in polynomial time when the height of the tree is a constant. Let T be the input tree having height t, i.e., T has $t+1$ levels $L_0, \ldots, L_t$.

L_0 contains only the root of T. Let $n_i = |L_i|$. We also assume that the nodes of L_i are ordered for all $i \geq 1$, i.e., $L_i = \{v_{i1}, \ldots, v_{in_i}\}$. For $j \leq l$, let $F(i,j,l)$ be the union of the induced subtrees of T rooted at the vertices $v_{ij}, \ldots, v_{il}$. We construct the tree $T(i,j,l)$ from $F(i,j,l)$ by connecting the roots of the subtrees to a common root.

Let feasible$(T(i,j,l), l_i, l_{i+1}, \ldots, l_t)$ be the function that decides if there is a feasible solution to CRMFC-T for the tree $T(i,j,l)$ by selecting at most l_m nodes from level m, where $i \leq m \leq t$. Note that computing the function feasible$(T = T(1,1,n_1), k_1, \ldots, k_t)$ solves the CRMFC-T problem. We consider the following recursive definition of feasible(). In the base case, if $i = t-1$, the function can be computed in polynomial time. Otherwise, if l_i is 0, let j' be the minimum index such that $v_{i+1,j'}$ is a child of v_{ij} and l' be the maximum index such that $v_{i+1,l'}$ is a child of v_{il}. In this case, feasible$(T(i,j,l), l_i, l_{i+1}, \ldots, l_t)$=feasible$(T(i+1, j', l'), l_{i+1}, \ldots, l_t)$. Otherwise, there must be a minimum index $j \leq j^1 \leq l$ such that a yes node v_{ij^1} is selected to be in the solution. For such a fixed $j < j^1 < l$, let j' be the minimum index such that $v_{i+1,j'}$ is a child of v_{ij} and l' be the maximum index such that $v_{i+1,l'}$ is a child of v_{i,j^1-1}. In this case, if there are values $l^1_{i+1}, \ldots, l^1_t, l^2_i, l^2_{i+1}, \ldots, l^2_t$ such that $l^2_i = l_i - 1$, $l_m = l^1_m + l^2_m$ for all $i+1 \leq m \leq t$, and both feasible$(T(i+1, j', l'), l^1_{i+1}, \ldots, l^1_t)$ and feasible$(T(i, j^1+1, l), l^2_i, l^2_{i+1}, \ldots, l^2_t)$ return yes, then feasible$(T(i,j,l), l_i, l_{i+1}, \ldots, l_t)$ also returns yes. Otherwise if for all j^1 there are no such values, feasible$(T(i,j,l), l_i, l_{i+1}, \ldots, l_t)$ returns no. The corner cases when $j^1 = j$ or $j^1 = l$ can be handled similarly.

It is not hard to verify that feasible$(T(i,j,l), l_i, l_{i+1}, \ldots, l_t)$ correctly decides whether there is a feasible solution or not for $T(i,j,l)$. To compute the feasible() function for all possible values one can use a simple dynamic programming based technique. In particular, one can store the values of the function for all possible parameters in a table. The table is filled up in a bottom-up manner, where the values corresponding to a level j subtree is computed before computations of the values corresponding to a level i subtree for $i < j$. It is not hard to see that the procedure would take polynomial time and space for a constant t.

5 Hardness of Approximation

In this section, we will prove the following theorem.

▶ **Theorem 15.** *For any constant c and any $\gamma \leq c^{n^c}$, NUkC under γ-PR is hard to approximate in polynomial time within a factor of γ, unless* NP = RP.

To prove this theorem, we use a chain of reductions that involves the following problems.

1-in-3SAT [28]

INSTANCE: An ordered pair (B, C) consisting of a set B of Boolean variables and a set C of clauses over B having three literals each in conjunctive normal form.

QUESTION: Is there a truth assignment for B such that every clause in C contains exactly one true literal?

RESOURCE MINIMIZATION FOR FIRE CONTAINMENT ON TREES (RMFC-T) [18, 23]

INSTANCE: A rooted tree T and an integer m.

QUESTION: Is there a set N of non-root nodes such that every root-leaf path contains a node from N and for any integer $j \geq 1$, $|N \cap L_j| \leq m$, where L_j is the set of nodes at distance exactly j from the root?

The chain of reductions that we use consists of the following reductions: (1) 3SAT to 1-in-3SAT, (2) 1-in-3SAT to RMFC-T, and (3) RMFC-T to NUkC. Note that NUkC under PR has a unique optimal solution. As we would like to show hardness for the PR version of

NUkC, we will consider "Unambiguous" version of all these problems. For "Unambiguous" version of 3SAT and 1-in-3SAT, if an instance has a feasible solution, the solution is unique. For "Unambiguous" version of RMFC-T, if an instance has a feasible solution, the solution has a specific structure that we will define shortly. For the reduction from 3SAT to 1-in-3SAT, we ensure that the reduction preserves the number of solutions. Such a reduction is called a parsimonious reduction. To refer to the Unambiguous version of a problem we add a prefix "U-" to the problem name. Next, we discuss the details of the reductions.

In a celebrated work, Valiant and Vazirani [29] showed that U-3SAT is hard, unless NP = RP. Schaefer [28] showed a reduction from 3SAT to 1-in-3SAT to prove the NP-hardness of the latter problem. As noted in [11] the reduction is parsimonious. We use the same reduction (now from U-3SAT to U-1-in-3SAT) to prove the hardness of U-1-in-3SAT, unless NP = RP.

Next, we discuss the reduction from 1-in-3SAT to RMFC-T. First, we define the Unambiguous version of RMFC-T. For a vertex v of a rooted tree T, let leaves(T_v) be the set of leaves at the subtree rooted at v. For any two feasible solutions S_1 and S_2 of RMFC-T, S_1 and S_2 are called equivalent, if the two sets $\cup_{v\in S_1}$ { leaves(T_v)} and $\cup_{v\in S_2}$ { leaves(T_v)} are identical. U-RMFC-T is same as RMFC-T except if the input instance has more than one feasible solutions, then all the feasible solutions are pairwise equivalent. The reduction from U-1-in-3SAT to U-RMFC-T appears in the appendix. The reduction is a non-trivial adaptation of the reduction due to Finbow et al. [18] from a version of 3SAT (RESTRICTED NAE 3-SAT) to the RMFC-T problem. We summarize our finding in the following lemma.

▶ **Lemma 16.** *Given a tree T, it is not possible to distinguish between the following two cases in polynomial time, unless* NP = RP.

- ***YES**: There is a solution to the* U-RMFC-T *instance with $m = 1$.*
- ***NO**: There is no solution to the* U-RMFC-T *instance with $m = 1$.*

To complete the chain of reductions, now we discuss the last reduction. In particular, we show a reduction from RMFC-T to NUkC that proves the following theorem.

▶ **Theorem 17.** *For any constant c and any $\gamma \leq c^{n^c}$, NUkC is* NP*-hard to approximate within a factor of γ in tree metrics.*

Note that this theorem has already been proved in [13]. However, it is not straightforward to show that the instances of NUkC they construct are perturbation resilient. Using a similar construction, we will argue that the instances of NUkC to which the instances of RMFC-T map are perturbation resilient. However, to ensure that the constructed instance of NUkC has a unique optimal solution, we will consider the Unambiguous version of RMFC-T.

5.1 Proof of Theorem 17

To prove the theorem we show a reduction from U-RMFC-T. As mentioned before, the reduction is similar to the reduction used by Chakrabarty et al. [13]. The construction is as follows. Let h be the height of the tree. We set P to be the leaves of the given tree T, i.e., $P = L_h$. For any edge (u, v) of T such that $u \in L_h$ and $v \in L_{h-1}$, assign a weight $(\gamma + 1)/2$ to (u, v). For any edge (u, v) of T such that $u \in L_i$ and $v \in L_{i-1}$ for $i \leq h - 1$, assign a weight $((\gamma + 1)^{h-i+1} - (\gamma + 1)^{h-i})/2$ to (u, v). Then the distance function d is the shortest-path metric on P induced by the weights of T. We set $t = h$, $r_t = 0$ and for any $1 \leq j < t$, $r_j = (\gamma + 1)^{t-j}$. Also $k_1 = \ldots = k_t = 1$. Now we have the following observation.

▶ **Observation 18.** *For any two leaves u, u' with a common ancestor $v \in L_j$, $d(u, u') \leq r_j$.*

Proof.

$$\begin{aligned} d(u,u') &\leq d(u,v) + d(v,u') \\ &= ((\gamma+1)/2 + ((\gamma+1)^2 - (\gamma+1))/2 + \ldots + ((\gamma+1)^{h-j} - (\gamma+1)^{h-j-1})/2) + \\ &\quad ((\gamma+1)/2 + ((\gamma+1)^2 - (\gamma+1))/2 + \ldots + ((\gamma+1)^{h-j} - (\gamma+1)^{h-j-1})/2) \\ &= r_j. \end{aligned}$$

◀

We note that the weight of any edge is bounded by $(\gamma+1)^{h-1} = c^{O(n^c h)}$ and thus can be represented using $O(n^c h)$ number of bits. It follows that the construction can be done in polynomial-time. We denote the constructed instance of NUkC by I. For simplicity, we use the terms point and leaf interchangeably. The following lemma completes the proof of Theorem 17 which follows from the construction and the fact that the feasible solutions for T are pairwise equivalent.

▶ **Lemma 19.** *If T is the "YES" case of Lemma 16, then the optimum dilation of I is 1. If T is the "NO" case of Lemma 16, then the optimum dilation of I is more than γ. Moreover, I has a unique optimal clustering.*

Proof. Let T be a "YES" instance and N be a solution for T. We construct a solution for I from N as follows. For any $v \in N$, let j be the integer such that $v \in L_j$. We select a leaf u from the subtree rooted at v and place a ball of radius r_j. We note that at most 1 ball of radius r_i is selected for all i, as $|N \cap L_i| \leq 1$. Now consider any point $w \in P$. Then there must be a node v in N along the path between w and the root. Let $v \in L_j$. Now the way we place the balls there must be a leaf u in the subtree rooted at v such that a ball of radius r_j is opened at u. As v is a common ancestor of u and w, from Observation 18, it follows that $d(u,w) \leq r_j$. Hence the ball $B(u, r_j)$ covers w.

Now let T be a "NO" instance and the optimum dilation of I be at most γ. Consider such a solution S corresponding to the instance I. We construct a solution N for U-RMFC-T on T using S as follows. For any $1 \leq j \leq t$, let u be the point where the ball (of radius at most γr_j) corresponding to r_j is placed. Let v be the ancestor of u that is in L_j. We add v to N. Note that, as S contains only one ball corresponding to the value r_i, $|N \cap L_i| \leq 1$ for all i. Now consider any leaf w. We show that N contains a node along the w-root path. Let B be a ball in S that covers w. Also let B be corresponding to the value r_j and is centered at the point u. Suppose v is the ancestor of u that is in L_j. As the radius of the ball at u is at most $\gamma r_j < r_{j-1}$, a point that is not contained in the subtree rooted at v cannot be covered by B. Hence w must be contained in the subtree rooted at v and thus w-root path contains $v \in N$. But this implies that N is a solution for T corresponding to the "YES" case and thus T must be a "YES" instance. But this is a contradiction and thus the optimum dilation of I must be more than γ.

As the feasible solutions for T are pairwise equivalent, it follows due to argument above that these feasible solutions get mapped to a unique optimal clustering of dilation 1. Similarly, the unique optimal clustering of dilation 1 gets mapped to a feasible solution of T. It follows that I has a unique optimal clustering. ◀

5.2 Hardness of Perturbation Resilient Version of NUkC

To show the hardness of the γ-perturbation-resilient version of NUkC, we prove that the constructed instances of U-NUkC in the reduction from U-RMFC-T to U-NUkC in tree metrics are γ-PR. First, we remind the reader of the tree metric d^* we used there. We are given a parameter γ and a tree T_γ with height h whose leaves are at the same distance from the root. The points in the metric space correspond to all the leaves of T_γ. Let n be the

number of leaves. Also, let L_i be the nodes of T_γ at level i for $1 \leq i \leq h$. For an edge (u, v) of T such that $u \in L_h$ and $v \in L_{h-1}$, we assign a weight $l(u, v) = (\gamma + 1)/2$ to (u, v). For each $u \in L_i$, $v \in L_{i-1}$ for $i \leq h - 1$ such that (u, v) is an edge in T_γ, we assign a weight $l(u, v) = ((\gamma + 1)^{h-i+1} - (\gamma + 1)^{h-i})/2$. For any two leaves w, w', $d^*(w, w')$ is the length of the shortest path between w and w', i.e., if the least common ancestor of w, w' is in L_j, then $d^*(w, w') = (\gamma + 1)^{h-j}$. We set $t = h$, $r_t = 0$ and for any $1 \leq j < t$, $r_j = (\gamma + 1)^{t-j}$. Also, $k_1 = \ldots = k_t = 1$. Let $L(\gamma)$ be the set of leaves of T_γ. As the distance between any two points and the r_j's are of the form $(\gamma + 1)^i$ for some i, we have the following observation.

▶ **Observation 20.** *The optimal dilation of the instance $\mathcal{I} = \{L(\gamma), d^*, t\}$ is $(\gamma + 1)^i$ for some integer $i \geq 0$.*

As we have shown before, for any constant c and any $\gamma \leq c^{n^c}$, U-NUkC is hard to approximate within a factor of γ for the metric space (T_γ, d^*), unless NP = RP. Next, we prove the following lemma.

▶ **Lemma 21.** *The instance $\mathcal{I} = \{L(\gamma), d^*, t\}$ is γ-PR.*

Proof. Let $\mathcal{O}$ be the optimal clustering of $\mathcal{I}$ and α be its dilation. Consider any γ-perturbation d' of d^*. We prove that the optimal clustering $\mathcal{O}'$ of the instance $\mathcal{I}' = \{L(\gamma), d', t\}$ is same as $\mathcal{O}$. Suppose for the sake of contradiction that $\mathcal{O}'$ is not same as $\mathcal{O}$. As d' is a γ-perturbation (the distances are non-increasing), the dilation of $\mathcal{O}'$ is at most α. We show that $\mathcal{O}'$ is also a feasible clustering for $\mathcal{I}$ with dilation at most α.

Consider any non-singleton cluster $C \in \mathcal{O}'$ with center c_1 that is covered by an r_j-ball for $j < t$. Then, for all pairs of points $p, q \in C$, $d'(p, q) \leq \alpha r_j$. This is true, as all the points are leaves of the tree. From Observation 20, it follows that $\alpha r_j = (\gamma + 1)^i$ for some i. As d' is a γ-perturbation of d^*, $d^*(p, q) \leq \gamma \cdot d'(p, q) < (\gamma + 1)^{i+1}$. Now, the way T_γ is constructed, there is no distance values strictly between $(\gamma + 1)^i$ and $(\gamma + 1)^{i+1}$. Hence, $d^*(p, q) \leq (\gamma + 1)^i = \alpha r_j$, and the ball $B(c_1, \alpha r_j)$ covers the points of the cluster C w.r.t. d^*. It follows that $\mathcal{O}'$ is also a feasible clustering for $\mathcal{I}$ with dilation at most α. But, as per our assumption $\mathcal{O}$ and $\mathcal{O}'$ are different, and thus the optimal clustering of $\mathcal{I}$ is not unique. This is a contradiction, and hence $\mathcal{O}$ and $\mathcal{O}'$ must be same. ◀

5.3 Hardness in Euclidean Metric

▶ **Theorem 22.** *For any constant κ and any $\beta \leq \kappa^{n^\kappa}$, NUkC under β-PR is hard to approximate within a factor of β in the Euclidean metric of dimension d for any $d \geq 1$, unless* NP = RP.

This result is in turn based on the following theorem due to Gupta [21].

▶ **Theorem 23** ([21])**.** *Any weighted tree T with L leaves can be embedded in polynomial-time into d-dimensional Euclidean space with $O(dL^{1/(d-1)} \min\{\log L, d\}^{1/2})$ distortion.*

The idea is to show that if there is a polynomial-time β-approximation for NUkC under β-PR in the Euclidean metric for any constant κ and any $\beta \leq \kappa^{n^\kappa}$, then there is also a polynomial-time γ-approximation for NUkC under γ-PR in tree metrics for any $\gamma \leq c^{n^c}$, where c is a constant. But, by Theorem 15 this is a contradiction, and hence the proof of the theorem follows. To obtain the γ-approximation in tree metrics we embed the tree metric into Euclidean metric of dimension d using the algorithm of Theorem 23. Then, we use the algorithm for Euclidean metric to obtain a solution for the embedded instance. Lastly, we map this solution back to the tree metric with sufficient expansion of the balls. For a suitable choice of β, one can show that the constructed solution is a γ-approximation. The details are given in the appendix.

References

1 David Adjiashvili, Andrea Baggio, and Rico Zenklusen. Firefighting on trees beyond integrality gaps. In *Proceedings of the 28th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain*, pages 2364–2383, 2017.

2 Manu Agarwal, Ragesh Jaiswal, and Arindam Pal. k-means++ under approximation stability. *Theor. Comput. Sci.*, 588:37–51, 2015.

3 Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and Euclidean k-median by primal-dual algorithms. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA*, pages 61–72, 2017.

4 Haris Angelidakis, Konstantin Makarychev, and Yury Makarychev. Algorithms for stable and perturbation-resilient problems. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada*, pages 438–451, 2017.

5 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Stability yields a PTAS for k-median and k-means clustering. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 309–318, 2010.

6 Pranjal Awasthi, Avrim Blum, and Or Sheffet. Center-based clustering under perturbation stability. *Inf. Process. Lett.*, 112(1-2):49–54, 2012.

7 Maria-Florina Balcan, Nika Haghtalab, and Colin White. k-center clustering under perturbation resilience. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy*, pages 68:1–68:14, 2016.

8 Maria-Florina Balcan and Yingyu Liang. Clustering under perturbation resilience. *SIAM J. Comput.*, 45(1):102–155, 2016.

9 Sayan Bandyapadhyay and Kasturi R. Varadarajan. Approximate clustering via metric partitioning. In *27th International Symposium on Algorithms and Computation, ISAAC 2016, Sydney, Australia*, pages 15:1–15:13, 2016.

10 Yonatan Bilu and Nathan Linial. Are stable instances easy? *Combinatorics, Probability & Computing*, 21(5):643–660, 2012.

11 Valentin Bura. A kernel method for positive 1-in-3-sat. *CoRR*, abs/1808.02821, 2018. `arXiv:1808.02821`.

12 Jaroslaw Byrka, Thomas Pensyl, Bartosz Rybicki, Aravind Srinivasan, and Khoa Trinh. An improved approximation for k-median and positive correlation in budgeted optimization. *ACM Trans. Algorithms*, 13(2):23:1–23:31, 2017.

13 Deeparnab Chakrabarty, Prachi Goyal, and Ravishankar Krishnaswamy. The non-uniform k-center problem. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, Rome, Italy*, pages 67:1–67:15, 2016.

14 Chandra Chekuri and Shalmoli Gupta. Perturbation resilient clustering for k-center and related problems via LP relaxations. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018 - Princeton, NJ, USA*, pages 9:1–9:16, 2018.

15 Julia Chuzhoy, Sudipto Guha, Eran Halperin, Sanjeev Khanna, Guy Kortsarz, Robert Krauthgamer, and Joseph Naor. Asymmetric k-center is $\log^* n$-hard to approximate. *J. ACM*, 52(4):538–551, 2005.

16 Vincent Cohen-Addad and Chris Schwiegelshohn. On the local structure of stable clustering instances. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, 2017*, pages 49–60, 2017.

17 Amit Deshpande, Anand Louis, and Apoorv Singh. On euclidean k-means clustering with alpha-center proximity. In Kamalika Chaudhuri and Masashi Sugiyama, editors, *Proceedings of Machine Learning Research*, volume 89 of *Proceedings of Machine Learning Research*, pages 2087–2095. PMLR, 16–18 apr 2019.

18 Stephen Finbow, Andrew D. King, Gary MacGillivray, and Romeo Rizzi. The firefighter problem for graphs of maximum degree three. *Discrete Mathematics*, 307(16):2094–2105, 2007.

19 Zachary Friggstad, Kamyar Khodamoradi, and Mohammad R. Salavatipour. Exact algorithms and lower bounds for stable instances of euclidean k-means. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2958–2972, 2019.

20 Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theor. Comput. Sci.*, 38:293–306, 1985.

21 Anupam Gupta. Embedding tree metrics into low-dimensional euclidean spaces. *Discrete & Computational Geometry*, 24(1):105–116, 2000.

22 Dorit S. Hochbaum and David B. Shmoys. A best possible heuristic for the k-center problem. *Math. Oper. Res.*, 10(2):180–184, 1985.

23 Andrew D. King and Gary MacGillivray. The firefighter problem for cubic graphs. *Discrete Mathematics*, 310(3):614–621, 2010.

24 Amit Kumar and Ravindran Kannan. Clustering with spectral norm and the k-means algorithm. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, Las Vegas, Nevada, USA*, pages 299–308, 2010.

25 Konstantin Makarychev, Yury Makarychev, and Aravindan Vijayaraghavan. Bilu-Linial stable instances of max cut and minimum multiway cut. In *Proceedings of the 25th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA*, pages 890–906, 2014.

26 Matús Mihalák, Marcel Schöngens, Rastislav Srámek, and Peter Widmayer. On the complexity of the metric TSP under stability considerations. In *Proceedings of 37th Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2011: Theory and Practice of Computer Science, Nový Smokovec, Slovakia.*, pages 382–393, 2011.

27 Rafail Ostrovsky, Yuval Rabani, Leonard J. Schulman, and Chaitanya Swamy. The effectiveness of Lloyd-type methods for the k-means problem. *J. ACM*, 59(6):28:1–28:22, 2012.

28 Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th Annual ACM Symposium on Theory of Computing, STOC 1978, San Diego, California, USA*, pages 216–226. ACM, 1978.

29 Leslie G Valiant and Vijay V Vazirani. NP is as easy as detecting unique solutions. In *Proceedings of the 17th Annual ACM Symposium on Theory of computing, STOC 1985*, pages 458–463. ACM, 1985.

A Examples demonstrating the definition of perturbation resilience

For more clarity, we describe the notion of ψ-perturbation-resilience in the context of NUkC using two examples in Figure 2 (top-left and bottom-left). In all our examples, the number of clusters $k = 2$ and the number of radii classes $t = 1$. For the instance shown at the top-left figure, let $r_1 = 10$. We claim that this instance is 2-perturbation-resilient. To see this note that here the optimal dilation is 1, and the optimal clusters are $\{a, c\}$ and $\{b, d\}$. Moreover, even if all the distances are perturbed by a factor of 2, the distance between a and c (resp. b and d) can be at most 10. Hence, the dilation of the previous clustering for the perturbed instance would be at most 1. But, as all the distances between a and b, a and d, c and b, and c and d are 50, in any 2-perturbation of the distances, the distance between the two points in any of these four pairs would be at least 25. Thus if both of the points in such a pair remain in same cluster, the dilation must be at least 2.5. As there is a clustering of dilation at most 1, in optimal clustering, both of these points cannot lie in the same cluster. Hence, the optimal clustering is unique and same as the one before. The top-right figure shows a 2-perturbed instance with the same optimal clustering. Now, consider the instance in the bottom-left figure. Let $r_1 = 15$. We claim that this instance is not 2-perturbation-resilient. To prove this we show a 2-perturbed instance where the optimal clustering is different. Note that in the original instance, the optimal dilation is 1, and the optimal clusters are $\{a, c\}$ and

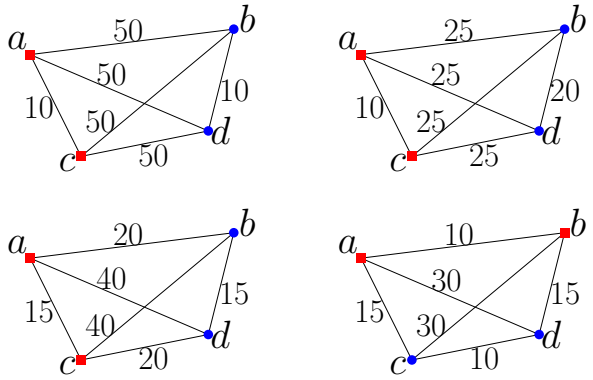


Figure 2 Examples demonstrating the definition of perturbation resilience. The top-right (resp. bottom-right) instance is a 2-perturbed instance of the top-left (resp. bottom-left) instance. The points in same optimal cluster are shown by same shape and color.

$\{b, d\}$. The 2-perturbed instance we consider is shown in the bottom-right figure. Note that in the perturbed instance the optimal clustering is $\{\{a, b\}, \{c, d\}\}$ with dilation 10/15=2/3. This is because any other clustering has a dilation at least 1.

B Reduction from 1-in-3SAT to RMFC-T

Finbow et al. [18] showed a reduction from Restricted NAE 3-SAT to RMFC-T. As per the definition of Restricted NAE 3-SAT, if the input instance has a feasible assignment, then it must at least have two. Thus, it cannot have a unique feasible solution. This is the reason behind our selection of the problem 1-in-3SAT, which can have a unique feasible solution. However, the reduction is motivated by the one in [18]. For consistency, we borrow some of their notations.

Given an instance I of 1-in-3SAT, we construct a rooted tree T with root r in multiple steps. Also, we choose the parameter $m = 1$. Before discussing the reduction, we have a few definitions to set up the stage. Throughout this discussion, we will use the operation *root* a copy of a rooted tree (T, r) at a vertex x of a graph G. This means we construct a new graph from the disjoint union of G and T by identifying x and r. A vertex v of a tree is said to be *defended* by a vertex u if the root to v path contains u. For any path, we assume that its root is one of the degree one vertices. Also, the length of a path is defined as the number of edges contained in it.

A *ladder tree* $\mathscr{L}^T(n)$ is a path having $2n + 1$ vertices such that the middle vertex of the path is identified as the root of the tree. See Figure 3(i). Thus, the root of $\mathscr{L}^T(n)$ has two branches each being a path of length n. A *bell tree* $\mathscr{B}^T(n, m)$ is formed by rooting a ladder tree $\mathscr{L}^T(n - m)$ at an endpoint of a path having m edges. The other endpoint of the path becomes the root of the bell tree. See Figure 3(ii). Thus, in the figure, the distance (in terms of edges) between a and b is m and the distance between a and a leaf is n. A *snake tree* $\mathscr{S}^T(n, m)$ is formed by rooting an $m - 1$ length path at the root of a bell tree $\mathscr{B}^T(n, m + 1)$. The root of the bell tree (or the path) becomes the root of the snake tree. Note that a snake tree has exactly one degree 3 vertex. See Figure 3(iii). Thus, in the figure, the distance between a and b is m, and the length of the path between a and a leaf such that the path contains b is n. A rooted tree T is called full if all leaves occur at the same level. A rooted tree T is called complete if every internal vertex has exactly two children. One simple observation is that a complete and full binary tree of height $h \geq 0$ has $2^{h+1} - 1$ vertices, and among those 2^h are leaves.

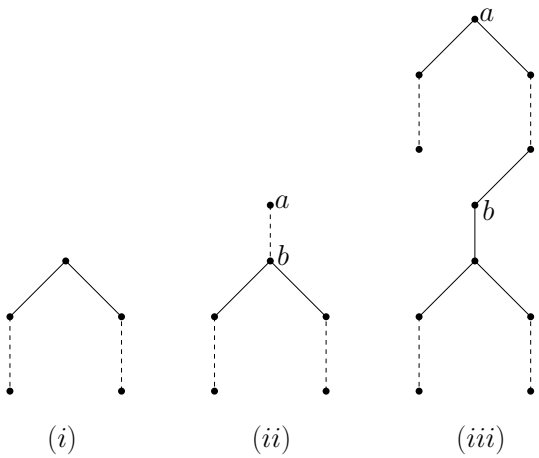


Figure 3 (i) A ladder tree. (ii) A bell tree. (iii) A snake tree. Dashed segments denote paths.

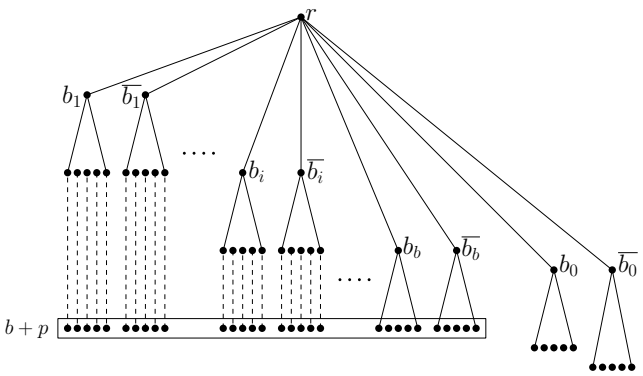


Figure 4 Figure showing the constructed tree after the first phase.

Now, we describe the construction. We are given the 1-in-3SAT instance $I=(B,C)$ with the set of variables $B = \{b_1, \ldots, b_b\}$ and the set of clauses $C = \{C_1, \ldots, C_n\}$. Let $p = \lceil \log n \rceil + 2$. Thus, $2^p \geq 4n$. We are going to construct a tree T which is initialized to the root vertex r. For each $1 \leq i \leq b$, root two paths of length i at the root r of T. Call the degree one vertices of these two paths b_i and $\overline{b_i}$. Root a complete and full binary tree of height p at b_i and $\overline{b_i}$ for each i. From each leaf of these trees root a path of length $b-i$. Call the leaves of these paths $t_{b_i,1}, \ldots, t_{b_i,2^p}$ and $t_{\overline{b_i},1}, \ldots, t_{\overline{b_i},2^p}$. Note that all the leaf nodes are now at a distance $b+p$ from r. Root two paths of length $b+1$ at r, and call the degree one vertices of these paths b_0 and $\overline{b_0}$. So far the construction is exactly the same as the one in [18]. In the following, we modify their construction to adapt it for our setting. From b_0 and $\overline{b_0}$ root a complete and full binary tree of height p and $p+1$, respectively, and call their leaves $t_{b_0,1}, \ldots, t_{b_0,2^p}$ and $t_{\overline{b_0},1}, \ldots, t_{\overline{b_0},2^{p+1}}$. This completes the first phase of the construction (see Figure 4).

In the second phase, we add clause gadgets by rooting special tree structures at the leaves of T constructed so far. For each $1 \leq j \leq n$, and for each literal l of C_j, root the snake tree $\mathscr{S}^T(4n+3, 4j-2)$ at $t_{\bar{l},j}$. For $1 \leq \tau \leq 3$, let q be the τ^{th} literal of C_j. Root $\mathscr{S}^T(4n+3, 4j-2+\tau)$ at $t_{q,j}$. Also, root the bell tree $\mathscr{B}^T(4n+3, 4j+1)$ at r for each $1 \leq j \leq n$. For $1 \leq i \leq 2^p$, add two children x_i and y_i of $t_{b_0,i}$. In each such added child,

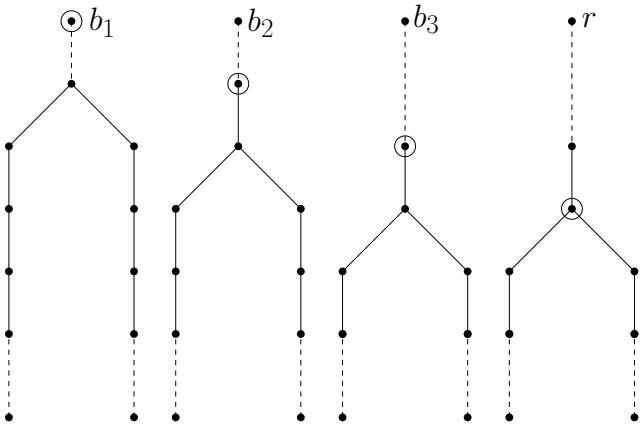


Figure 5 Figure showing parts of the three snake trees and the bell tree corresponding to the literals of C_j. The circled vertices are selected in the solution.

root the ladder tree $\mathscr{L}^T(4n+1)$. For $1 \leq i \leq 2^{p+1}$, add two children $\overline{x_i}$ and $\overline{y_i}$ of $t_{\overline{b_0},i}$. In each such added child, root the ladder $\mathscr{L}^T(4n)$. At each remaining leaf of T (as mentioned before) where no tree structure has been rooted so far, root $\mathscr{L}^T(4n+3)$. This completes the construction.

Now, let us give an intuitive description of the clause gadgets. Note that our main goal is to defend all the leaves. Consider the clause $C_j = (b_1 \vee b_2 \vee b_3)$. In a feasible solution, exactly one literal of C_j must be true, say b_1. Now suppose in the solution of U-RMFC-T we select the vertices corresponding to true literals, i.e., b_1, $\overline{b_2}$ and $\overline{b_3}$. Note that we have added one snake tree corresponding to each complemented literal of C_j. Thus, all the vertices in the snake trees corresponding to $\overline{b_2}$ and $\overline{b_3}$ are already defended. In this case, we can defend the degree three vertex (and all of its descendants) of the snake tree corresponding to $\overline{b_1}$ by choosing the degree three vertex itself. If more than one literal are true, then we need to defend vertices of at least two snake trees instead for which we would have to pick more than one vertices from a level. Now, we have also added three other snake trees one for each literal of C_j. As the snake tree corresponding to b_1 is already defended by b_1, we just need to defend the leaves of the remaining two. We can defend them by selecting the parent of the degree three vertex from the corresponding snake tree. In this way, we can also defend the last added bell tree by selecting its degree three vertex (see Figure 5). The alignments of these degree three vertices and their parents help us pick them in different levels. Note that if none of the literals are true, then we would need to defend the leaves of the three snake trees corresponding to the literals and in that case it is not possible to defend the leaves of the bell tree corresponding to C_j.

The argument behind the correctness of the reduction is similar to the one in [18]. The forward direction is simple. First, defend the vertices corresponding to true literals, i.e., if b_i (resp. $\overline{b_i}$) is true, defend b_i (resp. $\overline{b_i}$) at time i for $1 \leq i \leq b$. At time $b+1$, defend $\overline{b_0}$. From time $b+2$ to $b+p+1$, defend the unprotected descendant of b_0 which is not on the path from r to x_1. At time $b+p+2$, defend x_1. From time $b+p+3$ to $b+p+4n+3$, defend the tree greedily by picking a vertex at each level that contains the maximum number of nodes in the subtree rooted at it. The other direction is nontrivial, but similar counting arguments as in [18] should be used for the proof. It follows that the 1-in-3SAT formula is satisfiable if and only if all the leaves of T can be defended by selecting exactly one vertex from each level.

Finally, we show that all the feasible solutions are pairwise equivalent as claimed. This actually follows from the construction. Fix the unique feasible assignment to the 1-in-3SAT formula. Then, while finding a feasible solution for U-RMFC-T from the assignment in the above, in all the steps one need to select a unique vertex except when one needs to choose the parent of the degree three vertices of $\mathscr{S}^T(4n+3, 4j+1)$ and $\mathscr{B}^T(4n+3, 4j+1)$ both of which lie at the same level. However, irrespective of the selection, the set of leaves remains same. Thus, even though the solutions are different, the corresponding sets of leaves are same, and hence the solutions are pairwise equivalent.

C Hardness in Euclidean Metric

Let X and Y be two finite metric spaces with metrics d and d', respectively. Let $f : X \to Y$ be a map. Then, the *contraction* of f is defined as,

$$D_c(f) = \max_{x,y \in X} \frac{d(x,y)}{d'(f(x), f(y))}.$$

The *expansion* of f is similarly defined as,

$$D_e(f) = \max_{x,y \in X} \frac{d'(f(x), f(y))}{d(x,y)}.$$

The *distortion* of f, $D(f) = D_c(f) \cdot D_e(f)$. We need Theorem 23 due to Gupta [21] for proving the hardness result. Next, we prove Theorem 22.

Proof. Suppose there is a polynomial-time β-approximation for NUkC under β-PR in the Euclidean metric for any constant κ and any $\beta \leq \kappa^{n^\kappa}$. Then, we show that there is a polynomial-time γ-approximation for NUkC under γ-PR in tree metrics for any $\gamma \leq c^{n^c}$, where c is a constant. But, by Theorem 15 this is a contradiction, and hence the proof of the theorem follows.

Now, consider a constant c and any $\gamma \leq c^{n^c}$. Also, consider any instance of NUkC under γ-PR in the tree metric induced by the weighted tree T. We show how to get a γ-approximate solution for T using the approximation algorithm for the Euclidean metric. Let $\Delta = O(dn^{1/(d-1)} \log n)$. First, we embed the tree T into d-dimensional Euclidean space $\mathbb{R}^d$ using the algorithm of Theorem 23. Let $f : T \to \mathbb{R}^d$ be the embedding. Also, let d and d_f denote the tree and the Euclidean metric, respectively. We fix β such that $\beta \leq \gamma/\Delta$, and compute a β-approximate solution S of NUkC under β-PR for the Euclidean instance. Thereafter, we construct a solution S' for the problem on T from the solution S in the following way. For any node x of T, if S contains a ball centered at $f(x)$ with radius r, then we add the ball at x of radius $D_c(f) \cdot r$ to S', where $D_c(f)$ is the contraction of f. First, we show that the solution S' constructed in this way covers all the nodes of T. Consider any node x of T. Then, there is a ball in S centered at some point $f(y)$ that covers $f(x)$. Let r be the radius of this ball. It follows that S' contains the ball B centered at y having radius $D_c(f) \cdot r$. Now,

$$d(x,y) \leq D_c(f) \cdot d_f(f(x), f(y)) \leq D_c(f) \cdot r.$$

Hence, the ball B contains x, and thus S' is a feasible solution. Now, we show that the dilation $\alpha(S')$ of the balls in S' is at most γ times the optimum dilation. To this end, let OPT and OPT_f be the optimum dilation for the tree and the Euclidean instance, respectively.

Then, the dilation $\alpha(S')$ is at most $\beta \cdot OPT_f \cdot D_c(f)$. Now, as the distances between the points can get expanded by a factor of at most $D_e(f)$ due to the embedding, $OPT_f \leq D_e(f) \cdot OPT$. Here $D_e(f)$ is the expansion of f. Hence,

$$\alpha(S') \leq \beta \cdot D_e(f) \cdot OPT \cdot D_c(f) = \beta \cdot D(f) \cdot OPT \leq \beta \cdot \Delta \cdot OPT \leq \gamma \cdot OPT.$$

This completes the proof of the theorem. ◀

Low-Rank Binary Matrix Approximation in Column-Sum Norm

Fedor V. Fomin
Department of Informatics, University of Bergen, Norway
http://www.ii.uib.no/~fomin
fomin@ii.uib.no

Petr A. Golovach
Department of Informatics, University of Bergen, Norway
https://folk.uib.no/pgo041/
petr.golovach@ii.uib.no

Fahad Panolan[1]
Department of Computer Science and Engineering, IIT Hyderabad, India
https://iith.ac.in/~fahad/
fahad@cse.iith.ac.in

Kirill Simonov
Department of Informatics, University of Bergen, Norway
kirill.simonov@ii.uib.no

Abstract

We consider ℓ_1-Rank-r Approximation over GF(2), where for a binary $m \times n$ matrix $\mathbf{A}$ and a positive integer constant r, one seeks a binary matrix $\mathbf{B}$ of rank at most r, minimizing the column-sum norm $\|\mathbf{A}-\mathbf{B}\|_1$. We show that for every $\varepsilon \in (0,1)$, there is a randomized $(1+\varepsilon)$-approximation algorithm for ℓ_1-Rank-r Approximation over GF(2) of running time $m^{\mathcal{O}(1)}n^{\mathcal{O}(2^{4r}\cdot\varepsilon^{-4})}$. This is the first polynomial time approximation scheme (PTAS) for this problem.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Binary Matrix Factorization, PTAS, Column-sum norm

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.32

Category APPROX

Related Version A full version of the paper is available at https://arxiv.org/abs/1904.06141.

Funding *Fedor V. Fomin*: The research leading to these results have been supported by the Research Council of Norway via the project "MULTIVAL" (grant 263317). The work was done within the CEDAS center in Bergen.

1 Introduction

Low-rank matrix approximation is the method of compressing a matrix by reducing its dimension. It is the basic component of various methods in data analysis including Principal Component Analysis (PCA), one of the most popular and successful techniques used for dimension reduction in data analysis and machine learning [31, 15, 8]. In low-rank matrix approximation one seeks the best low-rank approximation of data matrix $\mathbf{A}$ with matrix $\mathbf{B}$ solving

$$\begin{aligned} &\text{minimize } \|\mathbf{A}-\mathbf{B}\|_\nu \\ &\text{subject to } \operatorname{rank}(\mathbf{B}) \leq r. \end{aligned} \tag{1}$$

[1] Corresponding author

© Fedor V. Fomin, Petr A. Golovach, Fahad Panolan, and Kirill Simonov;
licensed under Creative Commons License CC-BY

Here $\|\cdot\|_\nu$ is some matrix norm. The most popular matrix norms studied in the literature are the *Frobenius* $||\mathbf{A}||_F^2 = \sum_{i,j} a_{ij}^2$ and the *spectral* $\|\mathbf{A}\|_2 = \sup_{x\neq 0} \frac{\|\mathbf{Ax}\|_2}{\|\mathbf{x}\|_2}$ norms. By the Eckart-Young-Mirsky theorem [8, 27], (1) is efficiently solvable via Singular Value Decomposition (SVD) for these two norms. The spectral norm is an "extremal" norm – it measures the worst-case stretch of the matrix. On the other hand, the Frobenius norm is "averaging". Spectral norm is usually applied in the situation when one is interested in actual columns for the subspaces they define and is of greater interest in scientific computing and numerical linear algebra. The Frobenius norm is widely used in statistics and machine learning, see the survey of Mahony [24] for further discussions.

Recently there has been considerable interest in developing algorithms for low-rank matrix approximation problems for binary (categorical) data. Such variants of dimension reduction for high-dimensional data sets with binary attributes arise naturally in applications involving binary data sets, like latent semantic analysis [4], pattern discovery for gene expression[32], or web search models [19], see [7, 17, 14, 20, 30, 37] for other applications. In many such applications it is much more desirable to approximate a binary matrix $\mathbf{A}$ with a binary matrix $\mathbf{B}$ of small (GF(2) or Boolean) rank because it could provide a deeper insight into the semantics associated with the original matrix. There is a big body of work done on binary and Boolean low-rank matrix approximation, see [2, 3, 7, 22, 25, 26, 28, 35, 34] for further discussions.

Unfortunately, SVD is not applicable for the binary case which makes such problems computationally much more challenging. For a binary matrix, its squared Frobenius norm is equal to the number of its 1-entries, that is $\|\mathbf{A}\|_F^2 = \sum_{j=1}^{n}\sum_{i=1}^{m} a_{ij}$. Thus, the value $\|\mathbf{A}-\mathbf{B}\|_F^2$ measures the total Hamming distance from points (columns) of $\mathbf{A}$ to the subspace spanned by the columns of $\mathbf{B}$. For this variant of the low-rank binary matrix approximation, a number of approximation algorithms were developed, resulting in efficient polynomial time approximation schemes (EPTASes) obtained in [1, 9]. However, the algorithmic complexity of the problem for any vector-induced norm, including the spectral norm, remained open.

For binary matrices, the natural "extremal" norm to consider is the $\|\cdot\|_1$ norm, also known as *column-sum norm*, operator 1-norm, or Hölder matrix 1-norm. That is, for a matrix $\mathbf{A}$,

$$\|\mathbf{A}\|_1 = \sup_{\|\mathbf{x}\|_1\neq 0} \frac{\|\mathbf{Ax}\|_1}{\|\mathbf{x}\|_1} = \max_{1\leq j\leq n} \sum_{i=1}^{m} |a_{ij}|.$$

In other words, the column-sum norm is the maximum number of 1-entries in a column in $\mathbf{A}$, whereas the Frobenius norm is the total number of 1-entries in $\mathbf{A}$. The column-sum norm is analogous to the spectral norm, only it is induced by the ℓ_1 vector norm, not the ℓ_2 vector norm.

We consider the problem, where for an $m\times n$ binary data matrix $\mathbf{A}$ and a positive integer constant r, one seeks a binary matrix $\mathbf{B}$ optimizing

$$\begin{aligned} &\text{minimize } \|\mathbf{A}-\mathbf{B}\|_1 \\ &\text{subject to } \operatorname{rank}(\mathbf{B}) \leq r. \end{aligned} \tag{2}$$

Here, by the rank of the binary matrix $\mathbf{B}$ we mean its GF(2)-rank. We refer to the problem defined by (2) as to ℓ_1-Rank-r Approximation over GF(2). The value $\|\mathbf{A}-\mathbf{B}\|_1$ is the maximum Hamming distance from each of the columns of $\mathbf{A}$ to the subspace spanned by columns of $\mathbf{B}$ and thus, compared to approximation with the Frobenius norm, it could provide a more accurate dimension reduction.

It is easy to see by the reduction from the Closest String problem, that already for $r = 1$, ℓ_1-Rank-r Approximation over GF(2) is NP-hard. The main result of this paper is that (2) admits a polynomial time approximation scheme (PTAS). More precisely, we prove the following theorem.

▶ **Theorem 1.** *For every $\varepsilon \in (0, 1)$, there is a randomized $(1 + \varepsilon)$-approximation algorithm for ℓ_1-Rank-r Approximation over GF(2) of running time $m^{\mathcal{O}(1)} n^{\mathcal{O}(2^{4r} \cdot \varepsilon^{-4})}$.*

In order to prove Theorem 1 we obtain a PTAS for a more general problem, namely Binary Constrained k-Center. This problem has a strong expressive power and can be used to obtain PTASes for a number of problems related to ℓ_1-Rank-r Approximation over GF(2). For example, for the variant, when the rank of the matrix $\mathbf{B}$ is not over GF(2) but is Boolean. Or a variant of clustering, where we want to partition binary vectors into groups, minimizing the maximum distance in each of the group to some subspace of small dimension. We provide discussions of other applications of our work in Section 4.

Related work. The variant of (1) with both matrices $\mathbf{A}$ and $\mathbf{B}$ binary, and $\|\cdot\|_\nu$ being the Frobenius norm, is known as Low GF(2)-Rank Approximation. Due to numerous applications, various heuristic algorithms for Low GF(2)-Rank Approximation could be found in the literature [16, 17, 11, 20, 32].

When it concerns rigorous algorithmic analysis of Low GF(2)-Rank Approximation, Gillis and Vavasis [13] and Dan et al. [7] have shown that Low GF(2)-Rank Approximation is NP-complete for every $r \geq 1$. A subset of the authors studied parameterized algorithms for Low GF(2)-Rank Approximation in [10]. The first approximation algorithm for Low GF(2)-Rank Approximation is due to Shen et al. [32], who gave a 2-approximation algorithm for the special case of $r = 1$. For rank $r > 1$, Dan et al. [7] have shown that a $(r/2 + 1 + \frac{r}{2(2^r-1)})$-approximate solution can be formed from r columns of the input matrix $\mathbf{A}$. Recently, these algorithms were significantly improved in [1, 9], where efficient polynomial time approximation schemes (EPTASes) were obtained.

Also note that for general (non-binary) matrices a significant amount of work is devoted to $L1$-PCA, where one seeks a low-rank matrix $\mathbf{B}$ approximating given matrix $\mathbf{A}$ in *entrywise* ℓ_1 norm, see e.g. [33].

While our main motivation stems from low-rank matrix approximation problems, ℓ_1-Rank-r Approximation over GF(2) extends Closest String, very well-studied problem about strings. Given a set of binary strings $S = \{s_1, s_2, \ldots, s_n\}$, each of length m, the Closest String problem is to find the smallest d and a string s of length m which is within Hamming distance d to each $s_i \in S$.

A long history of algorithmic improvements for Closest String was concluded by the PTAS of running time $n^{\mathcal{O}(\epsilon^{-5})}$ by Li, Ma, and Wang [21], which running time was later improved to $n^{\mathcal{O}(\epsilon^{-2})}$ [23]. Let us note that Closest String can be seen as a special case of ℓ_1-Rank-r Approximation over GF(2) for $r = 1$. Indeed, Closest String is exactly the variant of ℓ_1-Rank-r Approximation over GF(2), where columns of $\mathbf{A}$ are strings of S and approximating matrix $\mathbf{B}$ is required to have all columns equal. Note that in a binary matrix $\mathbf{B}$ of rank 1 all non-zero columns are equal. However, it is easy to construct an equivalent instance of Closest String by attaching to each string of S a string 1^{m+1}, such that the solution to ℓ_1-Rank-r Approximation over GF(2) for $r = 1$ does not have zero columns.

Cygan et al. [6] proved that the existence of an EPTAS for Closest String, that is $(1 + \varepsilon)$-approximation in time $n^{\mathcal{O}(1)} \cdot f(\varepsilon)$, for any computable function f, is unlikely, as it would imply that FPT=W[1], a highly unexpected collapse in the hierarchy of parameterized

complexity classes. They also showed that the existence of a PTAS for Closest String with running time $f(\varepsilon)n^{o(1/\varepsilon)}$, for any computable function f, would contradict the Exponential Time Hypothesis. The result of Cygan et al. implies that ℓ_1-Rank-r Approximation over GF(2) also does not admit EPTAS (unless FPT=W[1]) already for $r = 1$.

A generalization of Closest String, k-closest strings is also known to admit a PTAS [18, 12]. This problem corresponds to the variant of ℓ_1-Rank-r Approximation over GF(2), where approximating matrix $\mathbf{B}$ is required to have at most k different columns. However, it is not clear how solution to this special case can be adopted to solve ℓ_1-Rank-r Approximation over GF(2).

1.1 Our approach

The usual toolbox of techniques to handle NP-hard variants of low-rank matrix approximation problems like sketching [36], sampling, and dimension reduction [5] is based on randomized linear algebra. It is very unclear whether any of these techniques can be used to solve even the simplest case of ℓ_1-Rank-r Approximation over GF(2) with $r = 1$. For example for sampling, the presence of just one outlier outside of a sample, makes all information we can deduce from the sample about the column sum norm of the matrix, completely useless. This is exactly the reason why approximation algorithms for Closest String do not rely on such techniques. On the other hand, randomized dimension reduction appears to be very helpful as a "preprocessing" procedure whose application allows us to solve ℓ_1-Rank-r Approximation over GF(2) by applying linear programming techniques similar to the ones developed for the Closest String. From a very general perspective, our algorithm consists of three steps. While each of these steps is based on the previous works, the way to combine these steps, as well as the correctness proof, is a non-trivial task. We start with a high-level description of the steps and then provide more technical explanations.

Step 1. In order to solve ℓ_1-Rank-r Approximation over GF(2), we encode it as the Binary Constrained k-Center problem. This initial step is almost identical to the encoding used in [9] for Low GF(2)-Rank Approximation. Informally, Binary Constrained k-Center is defined as follows. For a given set of binary vectors X, a positive integer k, and a set of constraints, we want to find k binary vectors $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ satisfying the constraints and minimizing $\max_{\mathbf{x}\in X} d_H(\mathbf{x}, C)$, where $d_H(\mathbf{x}, C)$ is the Hamming distance between $\mathbf{x}$ and the closest vector from C. For example, when $k = 1$ and there are no constraints, then this is just the Closest String problem over binary alphabet.

In the technical description below we give a formal definition of this encoding and in Section 4 we prove that ℓ_1-Rank-r Approximation over GF(2) is a special case of Binary Constrained k-Center. Now on, we are working with Binary Constrained k-Center.

Step 2. We give an approximate Turing reduction which allows to find a partition of vector set X into clusters $X_1, \ldots, X_k$ such that if we find a tuple of vectors $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ satisfying the constraints and minimizing $\max_{1\leq i\leq k, \mathbf{x}\in X_i} d_H(\mathbf{x}, \{\mathbf{c}_i\})$, then the same tuple C will be a good approximation to Binary Constrained k-Center. In order to obtain such a partition, we use the dimension reduction technique of Ostrovsky and Rabani [29]. While this provides us with important structural information, we are not done yet. Even with a given partition, the task of finding the corresponding tuple of "closest strings" C satisfying the constraints, is non-trivial.

Step 3. In order to find the centers, we implement the approach used by Li, Ma, and Wang in [21] to solve Closest String. By brute-forcing, it is possible to reduce the solution of the problem to special instances, which loosely speaking, have a large optimum. Moreover, Binary Constrained k-Center has an Integer Programming (IP) formulation. Similar to [21], for the reduced instance of Binary Constrained k-Center (which has a "large optimum") it is possible to prove that the randomized rounding of the corresponding Linear Program (LP) relaxation of this IP, provides a good approximation.

Now we give a more technical description of the algorithm.

Step 1. Binary Constrained k-Center. Note that the Binary Constrained k-Center problem is nearly identical to Binary Constrained Clustering defined in [9], except for the cost function. Still, for completeness we define Binary Constrained k-Center formally next. First, we need to define some notations. A k-ary relation R is a set of binary k-tuples with elements from $\{0,1\}$. A k-tuple $t=(t_1,\ldots,t_k)$ *satisfies* R, we write $t\in R$, if t is equal to one of the k-tuples in R.

▶ **Definition 2** (Vectors satisfying $\mathcal{R}$). *Let $\mathcal{R}=(R_1,\ldots,R_m)$ be a tuple of k-ary relations. We say that a tuple $C=(\mathbf{c}_1,\mathbf{c}_2,\ldots,\mathbf{c}_k)$ of binary m-dimensional vectors* satisfies *$\mathcal{R}$ and write $<C,\mathcal{R}>$, if $(\mathbf{c}_1[i],\ldots,\mathbf{c}_k[i])\in R_i$ for all $i\in\{1,\ldots,m\}$.*

For example, for $m=2$, $k=3$, $R_1=\{(0,0,1),(1,0,0)\}$, and $R_2=\{(1,1,1),(1,0,1),(0,0,1)\}$, the tuple of vectors

$$\mathbf{c}_1=\begin{pmatrix}0\\1\end{pmatrix},\ \mathbf{c}_2=\begin{pmatrix}0\\0\end{pmatrix},\ \mathbf{c}_3=\begin{pmatrix}1\\1\end{pmatrix}$$

satisfies $\mathcal{R}=(R_1,R_2)$ because $(\mathbf{c}_1[1],\mathbf{c}_2[1],\mathbf{c}_3[1])=(0,0,1)\in R_1$ and $(\mathbf{c}_1[2],\mathbf{c}_2[2],\mathbf{c}_3[2])=(1,0,1)\in R_2$.

Let us recall that the *Hamming distance* between two vectors $\mathbf{x},\mathbf{y}\in\{0,1\}^m$, where $\mathbf{x}=(x_1,\ldots,x_m)^\intercal$ and $\mathbf{y}=(y_1,\ldots,y_m)^\intercal$, is $d_H(\mathbf{x},\mathbf{y})=\sum_{i=1}^m|x_i-y_i|$ or, in words, the number of positions $i\in\{1,\ldots,m\}$ where x_i and y_i differ. Recall that for a set of vectors $C\subseteq\{0,1\}^m$ and a vector $\mathbf{x}\in\{0,1\}^m$, $d_H(\mathbf{x},C)=\min_{\mathbf{c}\in C}d_H(\mathbf{x},\mathbf{c})$. For sets $X,C\subset\{0,1\}^m$, we define $\mathrm{cost}(X,C)=\max_{\mathbf{x}\in X}d_H(\mathbf{x},C)$.

Now we define Binary Constrained k-Center formally.

Binary Constrained k-Center
Input: A set $X\subseteq\{0,1\}^m$ of n vectors, a positive integer k, and a tuple of k-ary relations $\mathcal{R}=(R_1,\ldots,R_m)$.
Task: Among all tuples $C=(\mathbf{c}_1,\ldots,\mathbf{c}_k)$ of vectors from $\{0,1\}^m$ satisfying $\mathcal{R}$, find a tuple C minimizing $\mathrm{cost}(X,C)$.

As in the case of Low GF(2)-Rank Approximation in [9], we prove that ℓ_1-Rank-r Approximation over GF(2) is a special case of Binary Constrained k-Center, where $k=2^r$. For completeness, this proof and other applications of Binary Constrained k-Center are given in Section 4. Thus, to prove Theorem 1, it is enough to design a PTAS for Binary Constrained k-Center.

▶ **Theorem 3.** *There is an algorithm for Binary Constrained k-Center that given an instance $J=(X,k,\mathcal{R})$ and $0<\varepsilon<1$, runs in time $m^{\mathcal{O}(1)}n^{\mathcal{O}((k/\varepsilon)^4)}$, and outputs a $(1+\varepsilon)$-approximate solution with probability at least $1-2n^{-2}$.*

By the argument above, Theorem 1 is an immediate corollary of Theorem 3.

Step 2: Dimension reduction. Let $J = (X, k, \mathcal{R} = (R_1, \ldots, R_m))$ be an instance of Binary Constrained k-Center and $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ be a solution to J, that is, a tuple of vectors satisfying $\mathcal{R}$. Then, the cost of C is $\mathrm{cost}(X, C)$. Given the tuple C, there is a natural way we can partition the set of vectors X into k parts $X_1 \uplus \cdots \uplus X_k$ such that

$$\mathrm{cost}(X, C) = \max_{i \in \{1,\ldots,k\}, \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i).$$

Thus, for each vector $\mathbf{x}$ in X_i, the closest to $\mathbf{x}$ vector from C is $\mathbf{c}_i$. We call such a partition $X_1 \uplus \cdots \uplus X_k$ the *clustering of X induced by C* and refer to the sets $X_1, \ldots, X_k$ as the *clusters corresponding to C*. We use $\mathsf{OPT}(J)$ to denote the cost of an optimal solution to J. That is, $\mathsf{OPT}(J) = \min\{\mathrm{cost}(X, C) \mid < C, \mathcal{R} >\}$. In fact, even if we know the clustering of X induced by a hypothetical optimal solution, finding a good solution is not trivial as the case when $k = 1$ is the same as the Closest String problem.

As mentioned before, our approach is to reduce to a version of Binary Constrained k-Center, where we know the partition of X, and solve the corresponding problem. That is, we design an approximation scheme for the following partitioned version of the problem.

Binary Constrained Partition Center
Input: A positive integer k, a set $X \subseteq \{0,1\}^m$ of n vectors partitioned into $X_1 \uplus \ldots \uplus X_k$, and a tuple of k-ary relations $\mathcal{R} = (R_1, \ldots, R_m)$.
Task: Among all tuples $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ of vectors from $\{0,1\}^m$ satisfying $\mathcal{R}$, find a tuple C minimizing $\max_{i \in \{1,\ldots,k\}, \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i)$.

For an instance $J' = (k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$ of Binary Constrained Partition Center, we use $\mathsf{OPT}(J')$ to denote the cost of an optimal solution to J'. That is,

$$\mathsf{OPT}(J') = \min_{C=(\mathbf{c}_1,\ldots,\mathbf{c}_k) \text{ s.t. } <C,\mathcal{R}>} \left\{ \max_{i \in \{1,\ldots,k\}, \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i) \right\}.$$

Clearly, for an instance $J = (X, k, \mathcal{R})$ of Binary Constrained k-Center and a partition of X into $X_1 \uplus \ldots \uplus X_k$, any solution to the instance $J' = (k, X = X_1 \uplus \ldots X_k, \mathcal{R})$ of Binary Constrained Partition Center, of cost d, is also a solution to J with cost at most d. We prove that there is a randomized polynomial time algorithm that given an instance $J = (X, k, \mathcal{R})$ of Binary Constrained k-Center and $0 < \epsilon \leq \frac{1}{4}$, outputs a collection $\mathcal{I}$ of Binary Constrained Partition Center instances $J' = (k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$ such that the cost of at least one instance in $\mathcal{I}$ is at most $(1 + 4\epsilon)\mathsf{OPT}(J)$ with high probability.

▶ **Lemma 4.** *There is an algorithm that given an instance $J = (X, k, \mathcal{R})$ of Binary Constrained k-Center, $0 < \epsilon \leq \frac{1}{4}$, and $\gamma > 0$, runs in time $m^2 n^{\mathcal{O}(k/\epsilon^4)}$, and outputs a collection $\mathcal{I}$ of $m \cdot n^{\mathcal{O}(k/\epsilon^4)}$ instances of Binary Constrained Partition Center such that each instance in $\mathcal{I}$ is of the form $(k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$, and there exists $J' \in \mathcal{I}$ such that $\mathsf{OPT}(J') \leq (1 + 4\epsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-\gamma}$.*

To prove Lemma 4, we use the dimension reduction technique of Ostrovsky and Rabani from [29]. Loosely speaking, this technique provides a linear map ψ with the following properties. For any $\mathbf{y} \in \{0,1\}^m$, $\psi(\mathbf{y})$ is a 0-1 vector of length $\mathcal{O}(\log n/\epsilon^4)$, and for any set Y of $n + k$ vectors, Hamming distances between any pair of vectors in $\psi(Y)$ are relatively preserved with high probability. So we assume that ψ is "a good map" for the set of vectors $X \cup C$, where $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ is a hypothetical optimal solution to J. Then, we guess the potential tuples of vectors $(\phi(\mathbf{c}_1), \ldots, \phi(\mathbf{c}_k))$ for the hypothetical optimal solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$, and use these choices for $(\phi(\mathbf{c}_1), \ldots, \phi(\mathbf{c}_k))$ to construct partitions of X, and thereby construct instances in $\mathcal{I}$. Lemma 4 is proved in Section 2.

Step 3: LP relaxation. Because of Lemma 4, to prove Theorem 3, it is enough to design a PTAS for Binary Constrained Partition Center which is more challenging part in our algorithm. So we prove the following lemma.

▶ **Lemma 5.** *There is an algorithm for Binary Constrained Partition Center that given an instance $J = (k, X = X_1^i \uplus \ldots \uplus X_k^i, \mathcal{R})$ and $0 < \epsilon < 1/2$, runs in time $m^{\mathcal{O}(1)} n^{\mathcal{O}((k/\epsilon)^4)}$, and outputs a solution of cost at most $(1+\epsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-2}$.*

Towards the proof of Lemma 5, we encode Binary Constrained Partition Center using an Integer programming (IP) formulation (see (6) in Section 3). We show that the randomized rounding using the solution of the linear programming relaxation of this IP provides a good approximation if the optimum value is large. Here we follow the approach similar to the one used by Li, Ma, and Wang in [21] to solve Closest String. We prove that there exist $Y_1 \subseteq X_1, \ldots, Y_k \subseteq X_k$, each of size $r = 1 + \frac{4}{\epsilon}$, with the following property. Let Q be the set of positions in $\{1, \ldots, m\}$ such that for each $i \in \{1, \ldots, k\}$ and $j \in Q$, all the vectors in Y_i agree at the position j, and for each $j \in Q$, $(\mathbf{y}_1[j], \ldots, \mathbf{y}_k[j]) \in R_j$, where $\mathbf{y}_i \in Y_i$ for all $i \in \{1, \ldots, k\}$. Then, for any solution of J such that for each $j \in Q$ the entries at the position j coincide with $(\mathbf{y}_1[j], \ldots, \mathbf{y}_k[j])$, the cost of this solution restricted to Q deviates from the cost of an optimal solution restricted to Q by at most $\frac{1}{r-1}\mathsf{OPT}(J)$. Moreover, the subproblem of J restricted to $\{1, \ldots, m\} \setminus Q$ has large optimum value and we could use linear programming to solve the subproblem. Lemma 5 is proved in Section 3.

Putting together. Next we explain how to prove Theorem 3 using Lemmata 4 and 5. Let $J = (X, k, \mathcal{R})$ be the input instance of Binary Constrained k-Center and $0 < \varepsilon < 1$ be the given error parameter. Let $\beta = \frac{\varepsilon}{8}$. Since $\varepsilon < 1$, $\beta < \frac{1}{4}$. Now, we apply Lemma 4 on J, β, and $\gamma = 2$. As a result, we get a collection $\mathcal{I}$ of instances of Binary Constrained Partition Center such that each instance in $\mathcal{I}$ is of the form $(k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$, and there exists $J' \in \mathcal{I}$ such that $\mathsf{OPT}(J') \leq (1 + 4\beta)\mathsf{OPT}(J)$ with probability at least $1 - n^{-2}$. From now on, we assume that this event happened. Next, for each instance $\widehat{J} \in \mathcal{I}$, we apply Lemma 5 with the error parameter β, and output the best solution among the solutions produced. Let $J' \in \mathcal{I}$ be the instance such that $\mathsf{OPT}(J') \leq (1+4\beta)\mathsf{OPT}(J) \leq (1+\frac{\varepsilon}{2})\mathsf{OPT}(J)$. Any solution to $\widehat{J} \in \mathcal{I}$ of cost d, is also a solution to J of cost at most d. Therefore, because of Lemmas 4 and 5, our algorithm outputs a solution of J with cost at most $(1+\beta)\mathsf{OPT}(J') = (1+\frac{\varepsilon}{8})(1+\frac{\varepsilon}{2})\mathsf{OPT}(J) \leq (1+\varepsilon)\mathsf{OPT}(J)$ with probability at least $1 - 2n^{-2}$, since both Lemmas 4 and 5 have the success probability of at least $1 - n^{-2}$. The running time of the algorithm follows from Lemmata 4 and 5.

As Theorem 3 is already proved using Lemmas 4 and 5, the rest of the paper is devoted to the proofs of Lemmata 4 and 5, and to the examples of the expressive power of Binary Constrained k-Center, including ℓ_1-Rank-r Approximation over GF(2). In Sections 2 and 3, we prove Lemmata 4 and 5, respectively. In Section 4, we give applications of Theorem 3.

2 Proof of Lemma 4

In this section we prove Lemma 4. The main idea is to map the given instance to a low-dimensional space while approximately preserving distances, then try all possible tuples of centers in the low-dimensional space, and construct an instance of Binary Constrained Partition Center by taking the optimal partition of the images with respect to a fixed tuple of centers back to the original vectors.

To implement the mapping, we employ the notion of (δ, ℓ, h)-distorted maps, introduced by Ostrovsky and Rabani [29]. Intuitively, a (δ, ℓ, h)-distorted map approximately preserves distances between ℓ and h, does not shrink distances larger than h too much, and does not expand distances smaller than ℓ too much. In what follows we make the definitions formal.

A metric space is a pair (P, d) where P is a set (whose elements are called points), and d is a distance function $d : P \times P \to \mathbb{R}$ (called a metric), such that for every $p_1, p_2, p_3 \in P$ the following conditions hold: (i) $d(p_1, p_2) \geq 0$, (ii) $d(p_1, p_2) = d(p_2, p_1)$, (iii) $d(p_1, p_2) = 0$ if and only if $p_1 = p_2$, and (iv) $d(p_1, p_2) + d(p_2, p_3) \geq d(p_1, p_3)$. Condition (iv) is called the triangle inequality. The pair $(\{0,1\}^m, d_H)$, binary vectors of lentgh m and the Hamming distance, is a metric space.

▶ **Definition 6** ([29]). *Let (P, d) and (P', d') be two metric spaces. Let $X, Y \subseteq P$. Let δ, ℓ, h be such that $\delta > 0$ and $h > \ell \geq 0$. A mapping $\psi : P \to P'$ is (δ, ℓ, h)-distorted on (X, Y) if and only if there exists $\alpha > 0$ such that for every $x \in X$ and $y \in Y$, the following conditions hold.*

1. *If $d(x, y) < \ell$, then $d(\psi(x), \psi(y)) < (1 + \delta)\alpha\ell$.*
2. *If $d(x, y) > h$, then $d(\psi(x), \psi(y)) > (1 - \delta)\alpha h$.*
3. *If $\ell \leq d(x, y) \leq h$, then $(1 - \delta)\alpha d(x, y) \leq d(\psi(x), \psi(y)) \leq (1 + \delta)\alpha d(x, y)$.*

If $X = Y$, then we say that ψ is (δ, ℓ, h)-distorted on X.

For any $r, r' \in \mathbb{N}$ and $\varepsilon > 0$, $\mathcal{A}_{r,r'}(\varepsilon)$ denotes a distribution over $r' \times r$ binary matrices $M \in \{0,1\}^{r' \times r}$, where entries are independent, identically distributed, random 0/1 variables with $\Pr[1] = \varepsilon$.

▶ **Proposition 7** ([29]). *Let $m, \ell \in \mathbb{N}$, and let $X \subseteq \{0,1\}^m$ be a set of n vectors. For every $0 < \epsilon \leq 1/2$, there exists a mapping $\phi : X \to \{0,1\}^{m'}$, where $m' = \mathcal{O}(\log n/\epsilon^4)$, which is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on X (with respect to the Hamming distance in both spaces). More precisely, for every $\gamma > 0$ there exists $\lambda > 0$, such that, setting $m' = \lambda \log n/\epsilon^4$, the linear map $\mathbf{x} \mapsto A\mathbf{x}$, where A is a random matrix drawn from $\mathcal{A}_{m,m'}(\epsilon^2/\ell)$, is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on X with probability at least $1 - n^{-\gamma}$.*

Now we are ready to prove Lemma 4. We restate it for convenience.

▶ **Lemma 4.** *There is an algorithm that given an instance $J = (X, k, \mathcal{R})$ of Binary Constrained k-Center, $0 < \epsilon \leq \frac{1}{4}$, and $\gamma > 0$, runs in time $m^2 n^{\mathcal{O}(k/\epsilon^4)}$, and outputs a collection $\mathcal{I}$ of $m \cdot n^{\mathcal{O}(k/\epsilon^4)}$ instances of Binary Constrained Partition Center such that each instance in $\mathcal{I}$ is of the form $(k, X = X_1 \uplus \ldots \uplus X_k, \mathcal{R})$, and there exists $J' \in \mathcal{I}$ such that $\mathsf{OPT}(J') \leq (1 + 4\epsilon)\mathsf{OPT}(J)$ with probability at least $1 - n^{-\gamma}$.*

Proof. Without loss of generality, we may assume $\mathsf{OPT}(J) > 0$. If $\mathsf{OPT}(J) = 0$, there are at most k distinct vectors in X, and we trivially construct a single instance of Binary Constrained Partition Center by grouping equal vectors together.

Let $n = |X|$ and $n' = n + k$. Let $\lambda = \lambda(\gamma)$ be the constant mentioned in Proposition 7, and $m' = \lambda \log n'/\epsilon^4$. Then, for each $\ell \in [m]$[2], we construct the collection $\mathcal{I}_\ell$ of $n^{\mathcal{O}(k/\epsilon^4)}$ Binary Constrained Partition Center instances as follows.

- Start with $\mathcal{I}_\ell := \emptyset$.
- Randomly choose a matrix A^ℓ from the distribution $\mathcal{A}_{m,m'}(\epsilon^2/\ell)$.

[2] For an integer $n \in \mathbb{N}$, we use $[n]$ as a shorthand for $\{1, \ldots, n\}$.

- For each choice of k vectors $\mathbf{c}'_1, \ldots, \mathbf{c}'_k \in \{0,1\}^{m'}$, construct a partition $X_1 \uplus \ldots \uplus X_k$ of X such that for each $\mathbf{x} \in X_i$, $\mathbf{c}'_i$ is one of the closest vectors to $A^\ell \mathbf{x}$ among $C' = \{\mathbf{c}'_1, \ldots, \mathbf{c}'_k\}$. Then, add $(k, X = X^i_1 \uplus \ldots X^i_k, \mathcal{R})$ to $\mathcal{I}_\ell$.

Finally, our algorithm outputs $\mathcal{I} = \bigcup_{\ell \in [m]} \mathcal{I}_\ell$ as the required collection of Binary Constrained Partition Center instances. Notice that for any $\ell \in [m]$, $|\mathcal{I}_\ell| = 2^{m'k} = n^{\mathcal{O}(k/\epsilon^4)}$. This implies that the cardinality of $\mathcal{I}$ is upper bounded by $m \cdot n^{\mathcal{O}(k/\epsilon^4)}$, and the construction of $\mathcal{I}_\ell$ takes time $m \cdot n^{\mathcal{O}(k/\epsilon^4)}$. Thus, the total running time of the algorithm is $m^2 \cdot n^{\mathcal{O}(k/\epsilon^4)}$.

Next, we prove the correctness of the algorithm. Let $\ell = \mathsf{OPT}(J)$ and $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ be an optimum solution of J. Let $Y_1, \ldots, Y_k$ be the clusters corresponding to C. Consider the step in the algorithm where we constructed $\mathcal{I}_\ell$. By Proposition 7, the map $\psi \colon \mathbf{x} \mapsto A^\ell \mathbf{x}$ is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on $X \cup C$ with probability at least $1 - n^{-\gamma}$. In the rest of the proof, we assume that this event happened. Let $\mathbf{c}'_1 = A^\ell \mathbf{c}_1, \ldots, \mathbf{c}'_k = A^\ell \mathbf{c}_k$. Consider the Binary Constrained Partition Center instance constructed for the choice of vectors $\mathbf{c}'_1, \ldots, \mathbf{c}'_k$. That is, let $X_1, \ldots, X_k$ be the partition of X such that for each $\mathbf{x} \in X_i$, $\mathbf{c}'_i$ is one of the closest vector to $A^\ell \mathbf{x}$ from $C' = \{\mathbf{c}'_1, \ldots, \mathbf{c}'_k\}$. Let J' be the instance $(k, X = X^i_1 \uplus \ldots X^i_k, \mathcal{R})$ of Binary Constrained Partition Center.

Now, we claim that C is a solution to J' with cost at most $(1+4\epsilon)\ell = (1+4\epsilon)\mathsf{OPT}(J)$. Since C satisfies $\mathcal{R}$, C is a solution of J'. To prove $\mathsf{OPT}(J') \leq (1+4\epsilon)\ell$, it is enough to prove that for each $i \in [k]$ and $\mathbf{x} \in X_i$, $d_H(\mathbf{x}, \mathbf{c}_i) \leq (1+4\epsilon)\ell$. Fix an index $i \in [k]$ and $\mathbf{x} \in X_i$. Suppose $\mathbf{x} \in Y_i$. Since C is an optimum solution of J with corresponding clusters $Y_1, \ldots Y_k$, we have that $d_H(\mathbf{y}, \mathbf{c}_i) \leq \ell$ for all $\mathbf{y} \in Y_i \cap X_i$. Thus, $d_H(\mathbf{x}, \mathbf{c}_i) \leq \ell$. So, now consider the case $\mathbf{x} \in Y_j$ for some $j \neq i$. Notice that if $d_H(\mathbf{x}, \mathbf{c}_i) \leq \ell$, then we are done. We have the following two subcases.

Case 1: $d_H(\mathbf{x}, \mathbf{c}_i) \leq \frac{\ell}{2\epsilon}$. We know that the map $\psi \colon \mathbf{x} \mapsto A^\ell \mathbf{x}$ is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on $X \cup C$, and let $\alpha > 0$ be the number such that conditions of Definition 6 hold. Since $\mathbf{x} \in X_i$, we have that (a) $d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_i)) \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_j))$. Since $d_H(\mathbf{x}, \mathbf{c}_j) \leq \ell$ (because $\mathbf{x} \in Y_j$) and ψ is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on $X \cup C$, we have that (b) $d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_j)) \leq (1+\epsilon)\alpha\ell$. Since $\ell < d_H(\mathbf{x}, \mathbf{c}_i) \leq \frac{\ell}{2\epsilon}$, and ψ is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on $X \cup C$, we have that (c) $(1-\epsilon)\alpha d_H(\mathbf{x}, \mathbf{c}_i) \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_i))$. The statements (a), (b), and (c) imply that

$$d_H(\mathbf{x}, \mathbf{c}_i) \leq \frac{1+\epsilon}{1-\epsilon}\ell \leq (1+4\epsilon)\ell,$$

where the last inequality holds since $\epsilon \leq 1/4$.

Case 2: $d_H(\mathbf{x}, \mathbf{c}_i) > \frac{\ell}{2\epsilon}$. We prove that this case is impossible by showing a contradiction. Since $\epsilon \leq 1/4$, in this case, we have that $d_H(\mathbf{x}, \mathbf{c}_i) > 2\ell$. Since ψ is $(\epsilon, \ell/4, \ell/2\epsilon)$-distorted on $X \cup C$, $d_H(\mathbf{x}, \mathbf{c}_i) > 2\ell$, and $d_H(\mathbf{x}, \mathbf{c}_j) \leq \ell$, we have that

$(1-\epsilon)\alpha \cdot 2\ell \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_i)) \leq d_H(\psi(\mathbf{x}), \psi(\mathbf{c}_j)) \leq (1+\epsilon)\alpha \cdot \ell.$

Then $2(1-\epsilon) \leq (1+\epsilon)$ and thus $\epsilon \geq 1/3$, which contradicts the assumption that $\epsilon \leq 1/4$. This completes the proof of the lemma. ◀

3 Proof of Lemma 5

For a set of positions $P \subset [m]$, let us define the Hamming distance restricted to P by

$$d^P_H(\mathbf{x}, \mathbf{y}) = \sum_{i \in P} |x_i - y_i|.$$

We use the following lemma in our proof.

▶ **Lemma 8.** *Let $Y = \{\mathbf{y}_1, \cdots, \mathbf{y}_l\} \subset \{0,1\}^m$ be a set of vectors and $\mathbf{c}^* \in \{0,1\}^m$ be a vector. Let $d^* = \mathrm{cost}(Y, \{\mathbf{c}^*\}) = \max_{\mathbf{y}\in Y} d_H(\mathbf{y}, \mathbf{c}^*)$. For any $r \in \mathbb{N}$, $r \geq 2$, there exist indices $i_1, \ldots, i_r$ such that for any $\mathbf{x} \in Y$*

$$d_H^P(\mathbf{x}, \mathbf{y}_{i_1}) - d_H^P(\mathbf{x}, \mathbf{c}^*) \leq \frac{1}{r-1} d^*,$$

where P is any subset of $Q_{i_1,\ldots,i_r}$ and $Q_{i_1,\ldots,i_r}$ is the set of positions where all of $\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_r}$ coincide (i.e., $Q_{i_1,\ldots,i_r} = \{j \in [m] \colon \mathbf{y}_{i_1}[j] = \mathbf{y}_{i_2}[j] = \ldots = \mathbf{y}_{i_r}[j]\}$).

Proof. For a vector $\mathbf{x} = \mathbf{y}_{\ell'} \in Y$ and $P \subseteq Q_{i_1,\ldots,i_r}$, let

$$\begin{aligned} J_P(\ell') &= \{j \in P \colon \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}, \text{ and} \\ J(\ell') &= \{j \in Q_{i_1,\ldots,i_r} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}. \end{aligned}$$

To prove the lemma it is enough to prove that $|J_P(\ell')| \leq \frac{1}{r-1} d^*$. Also, since $J_P(\ell') \subseteq J(\ell')$, to prove the lemma, it is enough to prove that $|J(\ell')| \leq \frac{1}{r-1} d^*$. Recall that for any $s \in [\ell]$ and $1 \leq i_1, \ldots, i_s \leq \ell$, $Q_{i_1,\ldots,i_s}$ is the set of positions where all of $\mathbf{y}_{i_1}, \ldots, \mathbf{y}_{i_s}$ coincide. For any $2 \leq s \leq r+1$ and $1 \leq i_1, \ldots, i_s \leq \ell$, let $p_{i_1,\ldots,i_s}$ be the number of mismatches between $\mathbf{y}_{i_1}$ and $\mathbf{c}^*$ at the positions in $Q_{i_1,\ldots,i_s}$. Let

$$\rho_s = \min_{1 \leq i_1,\ldots,i_s \leq n} \frac{p_{i_1,\ldots,i_s}}{d^*}.$$

Notice that for any $2 \leq s \leq r+1$, $\rho_s \leq 1$.

▷ Claim 9 (Claim 2.2 [21]). [3] For any s such that $2 \leq s \leq r$, there are indices $1 \leq i_1, i_2, \ldots, i_r \leq \ell$ such that for any $\mathbf{x} = \mathbf{y}_{\ell'} \in Y$, $|J(\ell')| \leq (\rho_s - \rho_{s+1}) d^*$.

Proof. Consider indices $1 \leq i_1, \ldots, i_s \leq \ell$ such that $p_{i_1,\ldots,i_s} = \rho_s \cdot d^*$. Next arbitrarily pick $r - s$ indices $i_{s+1}, i_{s+2}, \ldots, i_r$ from $[\ell] \setminus \{i_1, \ldots, i_s\}$. Next we prove that $i_1, i_2, \ldots, i_r$ are the required set of indices. Towards that, fix $\mathbf{x} = \mathbf{y}_{\ell'} \in Y$,

$$\begin{aligned} J(\ell') &= |\{j \in Q_{i_1,\ldots,i_r} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\ &\leq |\{j \in Q_{i_1,\ldots,i_s} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{x}[j] \text{ and } \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\ &\qquad\qquad \text{(Because } Q_{i_1,\ldots,i_r} \subseteq Q_{i_1,\ldots,i_s}\text{)} \\ &= |\{j \in Q_{i_1,\ldots,i_s} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\} \setminus \{j \in Q_{i_1,\ldots,i_s} \colon \mathbf{y}_{i_1}[j] = \mathbf{x}[j] \wedge \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\ &= |\{j \in Q_{i_1,\ldots,i_s} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\} \setminus \{j \in Q_{i_1,\ldots,i_s,\ell'} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \\ &\qquad\qquad \text{(Since } \mathbf{x} = \mathbf{y}_{\ell'}\text{)} \\ &= |\{j \in Q_{i_1,\ldots,i_s} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| - |\{j \in Q_{i_1,\ldots,i_s,\ell'} \colon \mathbf{y}_{i_1}[j] \neq \mathbf{c}^*[j]\}| \qquad (3) \\ &= p_{i_1,\ldots,i_s} - p_{i_1,\ldots,i_s,\ell'} \qquad \text{(By definition)} \\ &\leq (\rho_s - \rho_{s+1}) d^* \qquad (4) \end{aligned}$$

The equality (3) holds since $Q_{i_1,\ldots,i_s} \supseteq Q_{i_1,\ldots,i_s,\ell'}$. The inequality (4) holds because $p_{i_1,\ldots,i_s} = \rho_s \cdot d^*$ by the choice of $i_1, \ldots, i_s$, and $\rho_{s+1} d^* \leq p_{i_1,\ldots,i_s,\ell'}$ by definition. ◁

Notice that $(\rho_2 - \rho_3) + (\rho_3 - \rho_4) + \ldots + (\rho_r - \rho_{r+1}) = (\rho_2 - \rho_{r+1}) \leq \rho_2 \leq 1$. Thus, one of $(\rho_2 - \rho_3), (\rho_3 - \rho_4), \ldots, (\rho_r - \rho_{r+1})$ is at most $1/(r-1)$. This completes the proof. ◀

[3] We remark that Claim 2.2 in [21] is stated for a vector $\mathbf{c}$ such that $d^* = \mathrm{cost}(Y, \{\mathbf{c}\}) = \min_{\mathbf{c}'} \mathrm{cost}(Y, \{\mathbf{c}'\})$. But the steps of the same proof work in our case as well.

Consider the instance $J = (k, X = X_1^i \uplus \dots X_k^i, \mathcal{R})$ of Binary Constrained Partition Center. Let $C^* = (\mathbf{c}_1^*, \cdots, \mathbf{c}_k^*) \subset \{0,1\}^m$ be an optimum solution to J. Let $d_{opt} = \mathsf{OPT}(J) = \max_{i\in[k],\mathbf{x}\in X_i} d_H(\mathbf{x}, \mathbf{c}_i^*)$. For each $i \in [k]$ and $r \geq 2$, by Lemma 8, there exist r elements $\mathbf{x}_i^{(1)}, \dots, \mathbf{x}_i^{(r)}$ of X_i such that for any $\mathbf{x} \in X_i$,

$$d_H^P(\mathbf{x}, \mathbf{x}_i^{(1)}) - d_H^P(\mathbf{x}, \mathbf{c}_i^*) \leq \frac{1}{r-1} d_{opt}, \tag{5}$$

where P is any subset of Q_i, and Q_i is the set of coordinates on which $\mathbf{x}_i^{(1)}, \dots, \mathbf{x}_i^{(r)}$ agree. Let us denote as Q the intersection of all Q_i from which the positions not satisfying $\mathcal{R}$ are removed. That is,

$$Q = \left\{ j \in \bigcap_{i\in[k]} Q_i \colon (\mathbf{x}_1^{(1)}[j], \mathbf{x}_2^{(1)}[j], \dots, \mathbf{x}_k^{(1)}[j]) \in R_j \right\}.$$

Because of (5), there is an approximate solution where the coordinates $j \in Q$ are identified using $\mathbf{x}_1^{(1)}, \dots, \mathbf{x}_k^{(1)}$. Let $\overline{Q} = [m] \setminus Q$. Now the idea is to solve the problem restricted to $\overline{Q}$ separately, and then complement the solution on Q by the values of $\mathbf{x}_i^{(1)}$. We prove that for the "subproblem" restricted on $\overline{Q}$, the optimum value is *large*. Towards that we first prove the following lemma.

▶ **Lemma 10 (⋆).** [4] *Let $J = (k, X = X_1^i \uplus \dots X_k^i, \mathcal{R})$ be an instance of Binary Constrained Partition Center. Let $(\mathbf{c}_1^*, \dots, \mathbf{c}_k^*)$ be an optimal solution for J, and $r \geq 2$ be an integer. Then, there exist $\{\mathbf{x}_1^{(1)}, \dots, \mathbf{x}_1^{(r)}\} \subset X_1, \dots, \{\mathbf{x}_k^{(1)}, \dots, \mathbf{x}_k^{(r)}\} \subset X_k$ with the following properties. For each $i \in [k]$, let Q_i be the set of coordinates on which $\mathbf{x}_i^{(1)}, \dots, \mathbf{x}_i^{(r)}$ agree, $Q = \left\{ j \in \bigcap_{i\in[k]} Q_i \colon (\mathbf{x}_1^{(1)}[j], \mathbf{x}_2^{(1)}[j], \dots, \mathbf{x}_k^{(1)}[j]) \in R_j \right\}$, and $\overline{Q} = [m] \setminus Q$.*

- *For any $i \in [k]$ and $\mathbf{x} \in X_i$, $d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)}) - d_H^Q(\mathbf{x}, \mathbf{c}_i^*) \leq \frac{1}{r-1}\mathsf{OPT}(J)$, and*
- *$|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$.*

As mentioned earlier, we fix the entries of our solution in positions j of Q with values in $\mathbf{x}_1^{(1)}[j], \dots, \mathbf{x}_k^{(1)}[j]$. Towards finding the entries of our solution in positions of $\overline{Q}$, we define the following problem and solve it.

Binary Constrained Partition Center*
Input: A positive integer k, a set $X \subseteq \{0,1\}^m$ of n vectors partitioned into $X_1 \uplus \dots \uplus X_k$, a tuple of k-ary relations $\mathcal{R} = (R_1, \dots, R_m)$, and for all $\mathbf{x} \in X$, $d_{\mathbf{x}} \in \mathbb{N}$
Task: Among all tuples $C = (\mathbf{c}_1, \dots, \mathbf{c}_k)$ of vectors from $\{0,1\}^m$ satisfying $\mathcal{R}$, find a tuple C that minimizes the integer d such that for all $i \in [k]$ and $\mathbf{x} \in X_i$, $d_H(\mathbf{x}, \mathbf{c}_i) \leq d - d_{\mathbf{x}}$.

▶ **Lemma 11.** *Let $J' = (k, X = X_1 \uplus \dots X_k, \mathcal{R}, (d_{\mathbf{x}})_{\mathbf{x}\in X})$ be an instance of Binary Constrained Partition Center*, $\mathsf{OPT}(J') \geq \frac{m}{c}$ for some integer c, and $0 < \delta < 1/c$. Then, there is an algorithm which runs in time $m^{\mathcal{O}(1)} n^{\mathcal{O}(c^2 k/\delta^2)}$, and outputs a solution C of J', of cost at most $(1+\delta)\mathsf{OPT}(J')$ with probability at least $1 - n^{-2}$.*

Before proving Lemma 11, we explain how all these result put together to form a proof of Lemma 5. We restate Lemma 5 for the convenience of the reader.

[4] The proofs of results marked with ⋆ are deferred to the full version of the paper.

▶ **Lemma 5.** *There is an algorithm for* Binary Constrained Partition Center *that given an instance* $J = (k, X = X_1^i \uplus \ldots \uplus X_k^i, \mathcal{R})$ *and* $0 < \epsilon < 1/2$*, runs in time* $m^{\mathcal{O}(1)} n^{\mathcal{O}((k/\epsilon)^4)}$*, and outputs a solution of cost at most* $(1+\epsilon)\mathsf{OPT}(J)$ *with probability at least* $1 - n^{-2}$*.*

Proof. Let $J = (k, X = X_1^i \uplus \ldots X_k^i, \mathcal{R})$ be the input instance of Binary Constrained Partition Center, and $0 < \epsilon < \frac{1}{2}$ be the error parameter. Let $(\mathbf{c}_1^*, \ldots, \mathbf{c}_k^*)$ be an optimal solution for J. Let $r \geq 2$ be an integer which we fix later. First, for each $i \in [k]$ we obtain r vectors $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)} \in X_i$ which satisfy the conditions of Lemma 10. Their existence is guaranteed by Lemma 10, and we guess them in time $n^{\mathcal{O}(rk)}$ over all $i \in [k]$. For each $i \in [k]$, let Q_i be the set of coordinates on which $\mathbf{x}_i^{(1)}, \ldots, \mathbf{x}_i^{(r)}$ agree, $Q = \left\{ j \in \bigcap_{i \in [k]} Q_i \colon (\mathbf{x}_1^{(1)}[j], \mathbf{x}_2^{(1)}[j], \ldots, \mathbf{x}_k^{(1)}[j]) \in R_j \right\}$, and $\overline{Q} = [m] \setminus Q$. Next, we construct a solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ as follows. For each $i \in [k]$ and $j \in Q$, we set $\mathbf{c}_i[j] = \mathbf{x}_i^{(1)}[j]$.

Towards finding the entries of vectors $\mathbf{c}_1, \ldots, \mathbf{c}_k$ at the coordinates in $\overline{Q}$, we use Lemma 11. Let J' be the instance of Binary Constrained Partition Center*, which is a natural restriction of J to $\overline{Q}$. That is, $J' = (k, X' = X_1' \uplus \ldots X_k', \mathcal{R}|_{\overline{Q}}, (d_{\mathbf{x}|_{\overline{Q}}})_{\mathbf{x} \in X'})$, where for each $i \in [k]$, $X_i' = \{\mathbf{x}|_{\overline{Q}} \colon \mathbf{x} \in X_i\}$ and for each $\mathbf{x} \in X_i$, $d_{\mathbf{x}|_{\overline{Q}}} = d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)})$. By the second condition in Lemma 10, we have that $|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$.

▷ Claim 12. $\mathsf{OPT}(J) \leq \mathsf{OPT}(J') \leq \left(1 + \frac{1}{r-1}\right) \mathsf{OPT}(J)$.

Proof. First, we prove that $\mathsf{OPT}(J) \leq \mathsf{OPT}(J')$. Towards that we show that we can transform a solution $C' = (\mathbf{c}_1', \cdots, \mathbf{c}_k')$ of J' with the objective value d to a solution C of J with the same objective value. For each $i \in [k]$, consider $\widehat{\mathbf{c}}_i$ which is equal to $\mathbf{x}_i^{(1)}$ restricted to Q, and to $\mathbf{c}_i'$ restricted to $\overline{Q}$, and the solution $\widehat{C} = (\widehat{\mathbf{c}}_1, \cdots, \widehat{\mathbf{c}}_k)$. Clearly, $\widehat{C}$ satisfies $\mathcal{R}$ since on $\overline{Q}$ it is guaranteed by C' being a solution to J', and on Q by construction of Q. The objective value of C is

$$\begin{aligned}
\max_{i \in [k], \mathbf{x} \in X_i} d_H(\mathbf{x}, \mathbf{c}_i) &= \max_{i \in [k], \mathbf{x} \in X_i} \left(d_H^{\overline{Q}}(\mathbf{x}, \mathbf{c}_i) + d_H^Q(\mathbf{x}, \mathbf{c}_i) \right) \\
&= \max_{i \in [k], \mathbf{x} \in X_i} \left(d_H(\mathbf{x}|_{\overline{Q}}, \mathbf{c}_i') + d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)}) \right) \\
&= \max_{i \in [k], \mathbf{x} \in X_i} \left(d_H(\mathbf{x}|_{\overline{Q}}, \mathbf{c}_i') + d_{\mathbf{x}|_{\overline{Q}}} \right) = d.
\end{aligned}$$

Thus, $\mathsf{OPT}(J) \leq \mathsf{OPT}(J')$.

Next, we prove that $\mathsf{OPT}(J') \leq \left(1 + \frac{1}{r-1}\right) \mathsf{OPT}(J)$. Recall that $(\mathbf{c}_1^*, \ldots, \mathbf{c}_k^*)$ is an optimal solution for J. Then, $(\mathbf{e}_1^*, \ldots, \mathbf{e}_k^*)$, where each $\mathbf{e}_i^*$ is the restriction of $\mathbf{c}_i^*$ on $\overline{Q}$, is a solution for J'. For each $i \in [k]$ and $\mathbf{x} \in X_i$,

$$\begin{aligned}
d_H(\mathbf{x}|_{\overline{Q}}, \mathbf{e}_i^*) + d_{\mathbf{x}|_{\overline{Q}}} &= d_H^{\overline{Q}}(\mathbf{x}, \mathbf{c}_i^*) + d_H^Q(\mathbf{x}, \mathbf{x}_i^{(1)}) \\
&\leq d_H^{\overline{Q}}(\mathbf{x}, \mathbf{c}_i^*) + d_H^Q(\mathbf{x}, \mathbf{c}_i^*) + \frac{1}{r-1} \mathsf{OPT}(J) \qquad \text{(By Lemma 10)} \\
&\leq d_H(\mathbf{x}, \mathbf{c}_i^*) + \frac{1}{r-1} \mathsf{OPT}(J) \\
&\leq \left(1 + \frac{1}{r-1}\right) \mathsf{OPT}(J)
\end{aligned}$$

This completes the proof of the claim. ◁

Since $|\overline{Q}| \leq rk \cdot \mathsf{OPT}(J)$ and by Claim 12, we have that $\mathsf{OPT}(J') \geq \frac{|\overline{Q}|}{rk} = \frac{|\overline{Q}|}{c}$, where $c = rk$. Let $0 < \delta < \frac{1}{c}$ be a number which we fix later.

Now we apply Lemma 11 on the input J' and δ, and let $C' = (\mathbf{c}'_1, \ldots, \mathbf{c}'_k)$ be the solution for J' obtained. We know that the cost d' of $\mathbf{c}'$ is at most $(1+\delta)\mathsf{OPT}(J')$ with probability at least $1 - n^{-2}$. For the rest of the proof we assume that the cost $d' \leq (1+\delta)\mathsf{OPT}(J')$. Recall that we have partially computed the entries of the solution $\mathbf{c} = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ for the instance J. That is, for each $j \in Q$ and $i \in [k]$, we have already set the value of $\mathbf{c}_i[j]$. Notice that $C' \subseteq \{0,1\}^{|\overline{Q}|}$. Since J' is obtained from J by restricting to $\overline{Q}$, there is a natural bijection f from $\overline{Q}$ to $[|\overline{Q}|]$ such that for each $\mathbf{x} \in X$ and $j \in \overline{Q}$, $\mathbf{x}[j] = \mathbf{y}[f(j)]$, where $\mathbf{y} = \mathbf{x}|_{\overline{Q}}$. Now for each $i \in [k]$ and $j \in \overline{Q}$, we set $\mathbf{c}_i[j] = \mathbf{c}'_i[f(j)]$.

In Claim 12, we have proven that the solution C of J obtained in this way has cost at most d'. By Lemma 11, we know that $d' \leq (1+\delta)\mathsf{OPT}(J')$. By Claim 12, $\mathsf{OPT}(J') \leq (1 + \frac{1}{r-1})\mathsf{OPT}(J)$. Thus, we have that the cost of the solution C of J is at most $(1+\delta)(1 + \frac{1}{r-1})\mathsf{OPT}(J)$. Now we fix $r = (1 + \frac{4}{\epsilon})$ and $\delta = \frac{\epsilon}{(2\epsilon+8)k}$. Then the cost of C is at most $(1+\epsilon)\mathsf{OPT}(J)$.

Running time analysis. The number of choices for $\{\mathbf{x}_1^{(1)}, \ldots, \mathbf{x}_1^{(r)}\} \subset X_1, \ldots, \{\mathbf{x}_k^{(1)}, \ldots, \mathbf{x}_k^{(r)}\} \subset X_k$ is at most $n^{\mathcal{O}(rk)} = n^{\mathcal{O}(k/\epsilon)}$. For each such choice, we run the algorithm of Lemma 11 which takes time at most $m^{\mathcal{O}(1)} n^{\mathcal{O}(c^2 k/\delta^2)} = m^{\mathcal{O}(1)} n^{\mathcal{O}((k/\epsilon)^4)}$. Thus, the total running time is $m^{\mathcal{O}(1)} n^{\mathcal{O}((k/\epsilon)^4)}$. ◀

Now the only piece left is the proof of Lemma 11. We use the following tail inequality (a variation of Chernoff bound) in the proof of Lemma 11.

▶ **Proposition 13** (Lemma 1.2 [21]). *Let $X_1, \ldots, X_n$ be n independent 0-1 random variables, $X = \sum_{i=1}^{n} X_i$, and $0 < \epsilon \leq 1$. Then, $\Pr[X > E[X] + \epsilon n] \leq e^{-\frac{1}{3}n\epsilon^2}$.*

Finally, we prove Lemma 11.

Proof of Lemma 11. First, assume that $m < 9c^2 \log n/\delta^2$. If this is the case, we enumerate all possible solutions for J' and output the best solution. The number of solutions is at most $2^{k \cdot m} = n^{\mathcal{O}(c^2 k/\delta^2)}$. Thus, in this case the algorithm is exact and deterministic, and the running time bound holds. For the rest of the proof we assume that $m \geq 9c^2 \log n/\delta^2$.

Binary Constrained Partition Center* can be formulated as a 0-1 optimization problem as explained below. For each $j \in [m]$ and tuple $t \in R_j$, we use a 0-1 variable $y_{j,t}$ to indicate whether the j^{th} entries of a solution form a tuple $t \in R_j$ or not. For any $i \in [k]$, $\mathbf{x} \in X_i$, $j \in [m]$ and $t \in R_j$, denote $\chi_i(\mathbf{x}[j], t) = 0$ if $\mathbf{x}[j] = t[i]$ and $\chi_i(\mathbf{x}[j], t) = 1$ if $\mathbf{x}[j] \neq t[i]$. Now Binary Constrained Partition Center* can be defined as the following 0-1 optimization problem.

$$
\begin{aligned}
&\min d \\
&\text{subject to} \\
&\sum_{t \in R_j} y_{j,t} = 1, && \text{for all } j \in [m]; \qquad (6)\\
&\sum_{j \in [m]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y_{j,t} \leq d - d_{\mathbf{x}}, && \text{for all } i \in [k] \text{ and } \mathbf{x} \in X_i \\
&y_{j,t} \in \{0,1\}, && \text{for all } j \in [m] \text{ and } t \in R_j.
\end{aligned}
$$

Any solution $y_{j,t}$ ($j \in [m]$ and $t \in R_j$) to (6) corresponds to the solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ where for all $j \in [m]$ and $t \in R_j$ such that $y_{j,t} = 1$, we have $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) = t$.

Now, we solve the above optimization problem using linear programming relaxation and obtain a fractional solution $y^\star_{j,t}$ ($j \in [m]$ and $t \in R_j$) with cost d'. Clearly, $d' \leq d_{opt} = \mathsf{OPT}(J')$. Now, for each $j \in [m]$, independently with probability $y^\star_{j,t}$, we set $y'_{j,t} = 1$ and $y'_{j,t'} = 0$, for any $t' \in R_j \setminus \{t\}$. Then $y'_{j,t}$ ($j \in [m]$ and $t \in R_j$) form a solution to (6). Next we construct the solution $C = (\mathbf{c}_1, \ldots, \mathbf{c}_k)$ to Binary Constrained Partition Center*, corresponding to $y'_{j,t}$ ($j \in [m]$ and $t \in R_j$). That is, for all $j \in [m]$ and $t \in R_j$ such that $y_{j,t} = 1$, we have $(\mathbf{c}_1[j], \ldots, \mathbf{c}_k[j]) = t$.

For the running time analysis, notice that solving the linear program and performing the random rounding takes polynomial time in the size of the problem (6). And the size of (6) is polynomial in the size of J', so the running time bound is satisfied. It remains to show that the constructed solution has cost at most $(1+\delta)\mathsf{OPT}(J')$ with probability at least $1 - n^{-2}$.

For any $j \in [m]$, the above random rounding procedure ensures that there is exactly one tuple $t \in R_j$ such that $y'_{j,t} = 1$. This implies that for any $j \in [m]$, $i \in [k]$ and $\mathbf{x} \in X_i$, $\sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y'_{j,t}$ is a 0-1 random variable. Since for each $j \in [m]$ the rounding procedure is independent, we have that for any $i \in [k]$ and $\mathbf{x} \in X_i$ the random variables $(\sum_{t \in R_1} \chi_i(\mathbf{x}[1], t) \cdot y'_{1,t}), \ldots, (\sum_{t \in R_m} \chi_i(\mathbf{x}[m], t) \cdot y'_{j,t})$ are independent. Hence, for any $i \in [k]$ and $\mathbf{x} \in X_i$, the Hamming distance between $\mathbf{x}$ and $\mathbf{c}_i$, $d_H(\mathbf{x}, \mathbf{c}_i) = \sum_{j \in [m]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y'_{j,t}$, is the sum of m independent 0-1 random variables. For each $i \in [k]$ and $\mathbf{x} \in X_i$, we upper bound the expected value of $d_H(\mathbf{x}, \mathbf{c}_i)$ as follows.

$$\begin{aligned}
E[d_H(\mathbf{x}, \mathbf{c}_i)] &= E\left[\sum_{j \in [m]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y'_{j,t}\right] \\
&= \sum_{j \in [m]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot E[y'_{j,t}] \\
&= \sum_{j \in [m]} \sum_{t \in R_j} \chi_i(\mathbf{x}[j], t) \cdot y^\star_{j,t} \\
&\leq d' - d_{\mathbf{x}} \qquad \text{(By the constraints of (6))}
\end{aligned}$$

Fix $\epsilon = \frac{\delta}{c}$. Then, by Proposition 13, for all $i \in [k]$, and $\mathbf{x} \in X_i$,

$$\Pr[d_H(\mathbf{x}, \mathbf{c}_i) > d' - d_{\mathbf{x}} + \epsilon m] \leq e^{-\frac{1}{3} m \epsilon^2}.$$

Therefore, by the union bound,

$$\Pr[\text{There exist } i \in [k] \text{ and } \mathbf{x} \in X_i \text{ such that } d_H(\mathbf{x}, \mathbf{c}_i) > d' - d_{\mathbf{x}} + \epsilon m] \leq n \cdot e^{-\frac{1}{3} m \epsilon^2} \quad (7)$$

We remind that $m \geq 9c^2 \log n / \delta^2 = 9 \log n / \epsilon^2$ and so $n \cdot e^{-\frac{1}{3} m \epsilon^2} \leq n^{-2}$. Thus, by (7),

$$\Pr[\text{There exist } i \in [k] \text{ and } \mathbf{x} \in X_i \text{ such that } d_H(\mathbf{x}, \mathbf{c}_i) > d' - d_{\mathbf{x}} + \epsilon m] \leq n^{-2}. \quad (8)$$

Since $d' \leq \mathsf{OPT}(J')$ and $\mathsf{OPT}(J') \geq m/c$, $d' + \epsilon m \leq (1 + c\epsilon)\mathsf{OPT}(J')$. Then, the probability that there exist $i \in [k]$ and $\mathbf{x} \in X_i$ such that $d_H(\mathbf{x}, \mathbf{c}_i) > (1 + c\epsilon)\mathsf{OPT}(J') - d_{\mathbf{x}}$ is at most n^{-2} by (8). Since $c\epsilon = \delta$, the proof is complete. ◀

4 Applications

In this section we explain the impact of Theorem 3 about Binary Constrained k-Center to other problems around low-rank matrix approximation. We would like to mention that Binary Constrained k-Center is very similar to the Binary Constrained Clustering problem from [9]. In Binary Constrained k-Center we want to minimize the maximum

distance of a vector from the input set of vectors to the closest center, whereas in Binary Constrained Clustering the sum of distances is minimized. While these problems are different, the reduction we explain here, except a few details, are identical to the ones described in [9]. For reader's convenience, we give one reduction (Lemma 14) in full details and skip all other reductions, which are similar.

In the following lemma we show that ℓ_1-Rank-r Approximation over GF(2) is a special case of Binary Constrained k-Center.

▶ **Lemma 14** (⋆). *There is an algorithm that given an instance $(\mathbf{A}, r)$ of ℓ_1-Rank-r Approximation over GF(2), where $\mathbf{A}$ is an $m \times n$-matrix and r is an integer, runs in time $\mathcal{O}(m + n + 2^{2r})$, and outputs an instance $J = (X, k = 2^r, \mathcal{R})$ of Binary Constrained k-Center with the following property. Given any α-approximate solution C to J, an α-approximate solution $\mathbf{B}$ to $(\mathbf{A}, r)$ can be constructed in time $\mathcal{O}(rmn)$ and vice versa.*

Thus, Theorem 1 follows from Theorem 3 and Lemma 14.

Low Boolean-Rank Approximation. Let $\mathbf{A}$ be a binary $m \times n$ matrix. Now we consider the elements of $\mathbf{A}$ to be *Boolean* variables. The *Boolean rank* of $\mathbf{A}$ is the minimum r such that $\mathbf{A} = \mathbf{U} \wedge \mathbf{V}$ for a Boolean $m \times r$ matrix $\mathbf{U}$ and a Boolean $r \times n$ matrix $\mathbf{V}$, where the product is Boolean, that is, the logical $\wedge$ plays the role of multiplication and $\vee$ the role of sum. Here $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 1 = 1$, $0 \vee 0 = 0$, $0 \vee 1 = 1$, and $1 \vee 1 = 1$. Thus the matrix product is over the Boolean semi-ring $(0, 1, \wedge, \vee)$. This can be equivalently expressed as the normal matrix product with addition defined as $1 + 1 = 1$. Binary matrices equipped with such algebra are called *Boolean matrices*.

In Boolean ℓ_1-Rank-r Approximation, we are given an $m \times n$ binary data matrix $\mathbf{A}$ and a positive integer r, and we seek a binary matrix $\mathbf{B}$ optimizing

$$\begin{aligned} &\text{minimize } \|\mathbf{A} - \mathbf{B}\|_1 \\ &\text{subject to } \operatorname{rank}(\mathbf{B}) \leq r. \end{aligned}$$

Here, by the rank of binary matrix $\mathbf{B}$ we mean its Boolean rank, and norm $\|\cdot\|_1$ is the *column sum norm*. Similar to Lemma 14, one can prove that Boolean ℓ_1-Rank-r Approximation is a special case of Binary Constrained k-Center, where $k = 2^r$. Thus, we get the following corollary from Theorem 3.

▶ **Corollary 15.** *There is an algorithm for Boolean ℓ_1-Rank-r Approximation that given an instance $I = (\mathbf{A}, r)$ and $0 < \varepsilon < 1$, runs in time $m^{\mathcal{O}(1)} n^{\mathcal{O}(2^{4r}/\varepsilon^4)}$, and outputs a $(1 + \varepsilon)$-approximate solution with probability at least $1 - 2n^{-2}$.*

Projective k-center. The Binary Projective k-Center problem is a variation of the Binary k-Center problem, where the centers of clusters are linear subspaces of bounded dimension r. (For $r = 1$ this is Binary k-Center and for $k = 1$ this is ℓ_1-Rank-r Approximation over GF(2).) Formally, in Binary Projective k-Center we are given a set $X \subseteq \{0, 1\}^m$ of n vectors and positive integers k and r. The objective is to find a family of r-dimensional linear subspaces $C = \{C_1, \ldots, C_k\}$ over GF(2) minimizing $\max_{\mathbf{x} \in X} d_H(\mathbf{x}, \bigcup_{i=1}^{k} C)$.

To see that Binary Projective k-Center is a special case of Binary Constrained k-Center, we observe that the condition that C_i is an r-dimensional subspace over GF(2) can be encoded (as in Lemma 14) by 2^r constraints. This observation leads to the following lemma.

▶ **Lemma 16.** *There is an algorithm that given an instance (X, r, k) of* Binary Projective k-Center*, runs in time $\mathcal{O}(m + n + 2^{\mathcal{O}(rk)})$, and outputs an instance $J = (X, k' = 2^{kr}, \mathcal{R})$ of* Binary Constrained k-Center *with the following property. Given any α-approximate solution C to J, an α-approximate solution C' to (X, r, k) can be constructed in time $\mathcal{O}(rkmn)$ and vice versa.*

Combining Theorem 3 and Lemma 16 together, we get the following corollary.

▶ **Corollary 17.** *There is an algorithm for* Binary Projective k-Center *that given an instance $I = (X, r, k)$ and $0 < \varepsilon < 1$, where $X \subseteq \{0,1\}^m$ is a set of n vectors and $r, k \in \mathbb{N}$, runs in time $m^{\mathcal{O}(1)} n^{\mathcal{O}(2^{4kr}/\varepsilon^4)}$, and outputs a $(1+\varepsilon)$-approximate solution with probability at least $1 - 2n^{-2}$.*

5 Conclusion

In this paper we gave a randomized PTAS for the Binary Constrained k-Center problem. This yields the first approximation scheme for ℓ_1-Rank-r Approximation over GF(2) and its Boolean variant. This paper leaves several interesting open problems. The running time of our $(1+\varepsilon)$-approximation algorithm is $m^{\mathcal{O}(1)} n^{\mathcal{O}(2^{4r} \cdot \varepsilon^{-4})}$. How far is this running time from being optimal? A simple adaptation of the result of Cygan et al. [6] for Closest String, yields that already for $r = 1$, an $(1+\varepsilon)$-approximation in time $n^{\mathcal{O}(1)} \cdot f(\varepsilon)$, for any computable function f, would imply that FPT=W[1]. Also the existence of a PTAS for $r = 1$ with running time $f(\varepsilon) n^{o(1/\varepsilon)}$, for any computable function f, would contradict the Exponential Time Hypothesis [6]. But these results do not exclude the opportunity of having an algorithm of running time $f(r, \varepsilon) \cdot (nm)^{poly(1/\varepsilon)}$ for some function f. Even the existence of an algorithm for ℓ_1-Rank-r Approximation over GF(2) of running time $m^{\mathcal{O}(1)} n^{poly(r,\varepsilon)}$ is an interesting open question.

References

1. Frank Ban, Vijay Bhattiprolu, Karl Bringmann, Pavel Kolev, Euiwoong Lee, and David P. Woodruff. A PTAS for p-low rank approximation. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 747–766. SIAM, 2019. doi:10.1137/1.9781611975482.47.
2. Eduard Bartl, Radim Belohlávek, and Jan Konecny. Optimal decompositions of matrices with grades into binary and graded matrices. *Annals of Mathematics and Artificial Intelligence*, 59(2):151–167, June 2010. doi:10.1007/s10472-010-9185-y.
3. Radim Belohlávek and Vilém Vychodil. Discovery of optimal factors in binary data via a novel method of matrix decomposition. *J. Computer and System Sciences*, 76(1):3–20, 2010. doi:10.1016/j.jcss.2009.05.002.
4. Michael W Berry, Susan T Dumais, and Gavin W O'Brien. Using linear algebra for intelligent information retrieval. *SIAM review*, 37(4):573–595, 1995.
5. Avrim Blum, John Hopcroft, and Ravi Kannan. *Foundations of Data Science*. Cambridge University Press, 2020. URL: https://books.google.co.in/books?id=koHCDwAAQBAJ.
6. Marek Cygan, Daniel Lokshtanov, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. Lower Bounds for Approximation Schemes for Closest String. In *Proceedings of the 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, volume 53 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 12:1–12:10, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.SWAT.2016.12.
7. Chen Dan, Kristoffer Arnsfelt Hansen, He Jiang, Liwei Wang, and Yuchen Zhou. On low rank approximation of binary matrices. *CoRR*, abs/1511.01699, 2015. arXiv:1511.01699.

8 Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

9 Fedor V. Fomin, Petr A. Golovach, Daniel Lokshtanov, Fahad Panolan, and Saket Saurabh. Approximation schemes for low-rank binary matrix approximation problems. *CoRR*, abs/1807.07156, 2018. arXiv:1807.07156.

10 Fedor V. Fomin, Petr A. Golovach, and Fahad Panolan. Parameterized low-rank binary matrix approximation. In *Proceedings of the 45th International Colloquium on Automata, Languages, and Programming (ICALP)*, volume 107 of *LIPIcs*, pages 53:1–53:16. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.53.

11 Yinghua Fu, Nianping Jiang, and Hong Sun. Binary matrix factorization and consensus algorithms. In *Proceedings of the International Conference on Electrical and Control Engineering (ICECE)*, pages 4563–4567. IEEE, 2010.

12 Leszek Gasieniec, Jesper Jansson, and Andrzej Lingas. Approximation algorithms for hamming clustering problems. *J. Discrete Algorithms*, 2(2):289–301, 2004. doi:10.1016/S1570-8667(03)00079-0.

13 Nicolas Gillis and Stephen A. Vavasis. On the complexity of robust PCA and ℓ_1-norm low-rank matrix approximation. *CoRR*, abs/1509.09236, 2015. arXiv:1509.09236.

14 Harold W. Gutch, Peter Gruber, Arie Yeredor, and Fabian J. Theis. ICA over finite fields - separability and algorithms. *Signal Processing*, 92(8):1796–1808, 2012. doi:10.1016/j.sigpro.2011.10.003.

15 Harold Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

16 Peng Jiang and Michael T. Heath. Mining discrete patterns via binary matrix factorization. In *ICDM Workshops*, pages 1129–1136. IEEE Computer Society, 2013.

17 Peng Jiang, Jiming Peng, Michael Heath, and Rui Yang. A clustering approach to constrained binary matrix factorization. In *Data Mining and Knowledge Discovery for Big Data: Methodologies, Challenge and Opportunities*, pages 281–303. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

18 Yishan Jiao, Jingyi Xu, and Ming Li. On the k-closest substring and k-consensus pattern problems. In *Proceedings of the 15th Annual Symposium on Combinatorial Pattern (CPM)*, volume 3109 of *Lecture Notes in Comput. Sci.*, pages 130–144. Springer, 2004.

19 Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, 1999. doi:10.1145/324133.324140.

20 Mehmet Koyutürk and Ananth Grama. Proximus: A framework for analyzing very high dimensional discrete-attributed datasets. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 147–156, New York, NY, USA, 2003. ACM. doi:10.1145/956750.956770.

21 Ming Li, Bin Ma, and Lusheng Wang. On the closest string and substring problems. *J. ACM*, 49(2):157–171, 2002. doi:10.1145/506147.506150.

22 Haibing Lu, Jaideep Vaidya, Vijayalakshmi Atluri, and Yuan Hong. Constraint-aware role mining via extended boolean matrix decomposition. *IEEE Trans. Dependable Sec. Comput.*, 9(5):655–669, 2012. doi:10.1109/TDSC.2012.21.

23 Bin Ma and Xiaoming Sun. More efficient algorithms for closest string and substring problems. *SIAM J. Computing*, 39(4):1432–1443, 2009. doi:10.1137/080739069.

24 Michael W. Mahoney. Randomized algorithms for matrices and data. *Foundations and Trends in Machine Learning*, 3(2):123–224, 2011. doi:10.1561/2200000035.

25 Pauli Miettinen, Taneli Mielikäinen, Aristides Gionis, Gautam Das, and Heikki Mannila. The discrete basis problem. *IEEE Trans. Knowl. Data Eng.*, 20(10):1348–1362, 2008. doi:10.1109/TKDE.2008.53.

26 Pauli Miettinen and Jilles Vreeken. Model order selection for boolean matrix factorization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 51–59. ACM, 2011. doi:10.1145/2020408.2020424.

27 L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford Ser. (2)*, 11:50–59, 1960. doi:10.1093/qmath/11.1.50.

28 Barsha Mitra, Shamik Sural, Jaideep Vaidya, and Vijayalakshmi Atluri. A survey of role mining. *ACM Comput. Surv.*, 48(4):50:1–50:37, February 2016. doi:10.1145/2871148.

29 Rafail Ostrovsky and Yuval Rabani. Polynomial-time approximation schemes for geometric min-sum median clustering. *J. ACM*, 49(2):139–156, 2002. doi:10.1145/506147.506149.

30 Amichai Painsky, Saharon Rosset, and Meir Feder. Generalized independent component analysis over finite alphabets. *IEEE Trans. Information Theory*, 62(2):1038–1053, 2016. doi:10.1109/TIT.2015.2510657.

31 Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

32 Bao-Hong Shen, Shuiwang Ji, and Jieping Ye. Mining discrete patterns via binary matrix factorization. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 757–766, New York, NY, USA, 2009. ACM. doi:10.1145/1557019.1557103.

33 Zhao Song, David P. Woodruff, and Peilin Zhong. Low rank approximation with entrywise ℓ_1-norm error. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing (STOC)*, pages 688–701. ACM, 2017. doi:10.1145/3055399.3055431.

34 Jaideep Vaidya. Boolean matrix decomposition problem: Theory, variations and applications to data engineering. In *Proceedings of the 28th IEEE International Conference on Data Engineering (ICDE)*, pages 1222–1224. IEEE Computer Society, 2012. doi:10.1109/ICDE.2012.144.

35 Jaideep Vaidya, Vijayalakshmi Atluri, and Qi Guo. The role mining problem: finding a minimal descriptive set of roles. In *Proceedings of the 12th ACM Symposium on Access Control Models and (SACMAT)*, pages 175–184, 2007. doi:10.1145/1266840.1266870.

36 David P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1–2):1–157, 2014. doi:10.1561/0400000060.

37 Arie Yeredor. Independent component analysis over Galois fields of prime order. *IEEE Trans. Information Theory*, 57(8):5342–5359, 2011. doi:10.1109/TIT.2011.2145090.

Pinning down the Strong Wilber 1 Bound for Binary Search Trees

Parinya Chalermsook
Aalto University, Finland
chalermsook@gmail.com

Julia Chuzhoy
Toyota Technological Institute at Chicago, IL, USA
cjulia@ttic.edu

Thatchaphol Saranurak
Toyota Technological Institute at Chicago, IL, USA
saranurak@ttic.edu

Abstract

The dynamic optimality conjecture, postulating the existence of an $O(1)$-competitive online algorithm for binary search trees (BSTs), is among the most fundamental open problems in dynamic data structures. Despite extensive work and some notable progress, including, for example, the Tango Trees (Demaine et al., FOCS 2004), that give the best currently known $O(\log\log n)$-competitive algorithm, the conjecture remains widely open. One of the main hurdles towards settling the conjecture is that we currently do not have approximation algorithms achieving better than an $O(\log\log n)$-approximation, even in the offline setting. All known non-trivial algorithms for BST's so far rely on comparing the algorithm's cost with the so-called Wilber's first bound (WB-1). Therefore, establishing the worst-case relationship between this bound and the optimal solution cost appears crucial for further progress, and it is an interesting open question in its own right.

Our contribution is two-fold. First, we show that the gap between the WB-1 bound and the optimal solution value can be as large as $\Omega(\log\log n/\log\log\log n)$; in fact, we show that the gap holds even for several stronger variants of the bound. Second, we provide a simple algorithm, that, given an integer $D > 0$, obtains an $O(D)$-approximation in time $\exp\left(O\left(n^{1/2^{\Omega(D)}}\log n\right)\right)$. In particular, this yields a constant-factor approximation algorithm with sub-exponential running time. Moreover, we obtain a simpler and cleaner efficient $O(\log\log n)$-approximation algorithm that can be used in an online setting. Finally, we suggest a new bound, that we call the *Guillotine Bound*, that is stronger than WB-1, while maintaining its algorithm-friendly nature, that we hope will lead to better algorithms. All our results use the geometric interpretation of the problem, leading to cleaner and simpler analysis.

2012 ACM Subject Classification Theory of computation → Data structures design and analysis

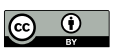
Keywords and phrases Binary search trees, Dynamic optimality, Wilber bounds

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.33

Category APPROX

Related Version A full version of the paper is available at `https://arxiv.org/abs/1912.02900`.

Funding *Parinya Chalermsook*: Supported by European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 759557) and by Academy of Finland Research Fellows, under grant No. 310415.
Julia Chuzhoy: Supported in part by NSF grant CCF-1616584. Part of the work was done while the second author was a Weston visiting professor at the Department of Computer Science and Applied Mathematics, Weizmann Institute of Science.



© Parinya Chalermsook, Julia Chuzhoy, and Thatchaphol Saranurak;
licensed under Creative Commons License CC-BY



1 Introduction

Binary search trees (BST's) are a fundamental data structure that has been extensively studied for many decades. Informally, suppose we are given as input an **online** access sequence $X = \{x_1, \ldots, x_m\}$ of keys from $\{1, \ldots, n\}$, and our goal is to maintain a binary search tree T over the set $\{1, \ldots, n\}$ of keys. The algorithm is allowed to modify the tree T after each access; the tree obtained after the ith access is denoted by T_{i+1}. Each such modification involves a sequence of *rotation* operations that transform the current tree T_i into a new tree T_{i+1}. The cost of the transformation is the total number of rotations performed plus the depth of the key x_i in the tree T_i. The total cost of the algorithm is the total cost of all transformations performed as the sequence X is processed. We denote by $\mathsf{OPT}(X)$ the smallest cost of any algorithm for maintaining a BST for the access sequence X, when the whole sequence X is known to the algorithm in advance.

Several algorithms for BST's, whose costs are guaranteed to be $O(m \log n)$ for any access sequence, such as AVL-trees [1] and red-black trees [2], are known since the 60's. Moreover, it is well known that there are length-m access sequences X on n keys, for which $\mathsf{OPT}(X) = \Omega(m \log n)$. However, such optimal worst-case guarantees are often unsatisfactory from both practical and theoretical perspectives, as one can often obtain better results for "structured" inputs. Arguably, a better notion of the algorithm's performance to consider is *instance optimality*, where the algorithm's performance is compared to the optimal cost $\mathsf{OPT}(X)$ for the specific input access sequence X. This notion is naturally captured by the algorithm's *competitive ratio*: we say that an algorithm for BST's is α-*competitive*, if, for every online input access sequence X, the cost of the algorithm's execution on X is at most $\alpha \cdot \mathsf{OPT}(X)$. Since for every length-m access sequence X, $\mathsf{OPT}(X) \geq m$, the above-mentioned algorithms that provide worst-case $O(m \log n)$-cost guarantees are also $O(\log n)$-competitive. However, there are many known important special cases, in which the value of the optimal solution is $O(m)$, and for which the existence of an $O(1)$-competitive algorithm would lead to a much better performance, including some interesting applications, such as, for example, adaptive sorting [23, 6, 19, 22, 13, 20, 12, 8, 7, 3, 5, 4]. A striking conjecture of Sleator and Tarjan [21] from 1985, called the *dynamic optimality conjecture*, asserts that the *Splay Trees* provide an $O(1)$-competitive algorithm for BST's. This conjecture has sparked a long line of research, but despite the continuing effort, and the seeming simplicity of BST's, it remains widely open. In a breakthrough result, Demaine et al. [10] proposed the Tango Trees algorithm, that achieves an $O(\log \log n)$-competitive ratio, and has remained the best known algorithm for the problem, for over 15 years. A natural avenue for overcoming this barrier is to first consider the "easier" task of designing (offline) approximation algorithms, whose approximation factor is below $O(\log \log n)$. Designing better approximation algorithms is often a precursor to obtaining better online algorithms, and it is a natural stepping stone towards this goal.

The main obstacle towards designing better algorithms, both in the online and the offline settings, is obtaining tight lower bounds on the value $\mathsf{OPT}(X)$, that can be used in algorithm design. If the input access sequence X has length m, and it contains n keys, then it is easy to see that $\mathsf{OPT}(X) \geq \Omega(m)$, and, by using any balanced BST's, such as AVL-trees, one can show that $\mathsf{OPT}(X) = O(m \log n)$. This trivially implies an $O(\log n)$-approximation for both offline and online settings. However, in order to obtain better approximation, these simple bounds do not seem sufficient. Wilber [25] proposed two new bounds, that we refer to as the first Wilber Bound (WB-1) and the second Wilber Bound (WB-2). He proved that, for every input sequence X, the values of both these bounds on X are at most $\mathsf{OPT}(X)$.

The breakthrough result of Demaine et al. [10], that gives an $O(\log\log n)$-competitive online algorithm, relies on the WB-1 bound. In particular, they show that the cost of the solution produced by their algorithm is within an $O(\log\log n)$-factor from the WB-1 bound on the given input sequence X, and hence from $\mathsf{OPT}(X)$. This in turn implies that, for every input sequence X, the value of the WB-1 bound is within an $O(\log\log n)$ factor from $\mathsf{OPT}(X)$. Follow-up work [24, 14] improved several aspects of Tango Trees, but it did not improve the approximation factor. Additional lower bounds on OPT, that subsume both the WB-1 and the WB-2 bounds, were suggested in [9, 11, 15], but unfortunately it is not clear how to exploit them in algorithm design. To this day, the only method we have for designing non-trivial online or offline approximation algorithms for BST's is by relying on the WB-1 bound, and this seems to be the most promising approach for obtaining better algorithms. In order to make further progress on both online and offline approximation algorithms for BST's, it therefore appears crucial that we better understand the relationship between the WB-1 bound and the optimal solution cost.

Informally, the WB-1 bound relies on recursive partitioning of the input key sequence, that can be represented by a partitioning tree. The standard WB-1 bound (that we refer to as the *weak* WB-1 bound) only considers a single such partitioning tree. It is well-known (see e.g. [10, 24, 16]), that the gap between $\mathsf{OPT}(X)$ and the weak WB-1 bound for an access sequence X may be as large as $\Omega(\log\log n)$. However, the "bad" access sequence X used to obtain this gap is highly dependent on the fixed partitioning tree T. It is then natural to consider a stronger variant of WB-1, that we refer to as *strong* WB-1 bound and denote by $\mathsf{WB}(X)$, that maximizes the weak WB-1 bound over all such partitioning trees. As suggested by Iacono [16], and by Kozma [17], this gives a promising approach for improving the $O(\log\log n)$-approximation factor.

In this paper, we show that, even for this strong variant of Wilber Bound, the gap between $\mathsf{OPT}(X)$ and $\mathsf{WB}(X)$ may be as large as $\Omega(\log\log n/\log\log\log n)$. This negative result extends to an even stronger variant of the Wilber Bound that we discuss below.

Our second set of results is algorithmic. We show an (offline) algorithm that, given an input sequence X and a positive integer D, obtains an $O(D)$-approximation, in time $\mathrm{poly}(m)\cdot\exp\left(n^{1/2^{\Omega(D)}}\log n\right)$. When D is constant, the algorithm obtains an $O(1)$-approximation in sub-exponential time. When D is $\Theta(\log\log n)$, it matches the best current efficient $O(\log\log n)$-approximation algorithm. In the latter case, we can also adapt the algorithm to the online setting, obtaining an $O(\log\log n)$-competitive online algorithm.

All our results use the geometric interpretation of the problem, introduced by Demaine et al. [9], leading to clean divide-and-conquer-style arguments that avoid, for example, the notion of pointers and rotations. We feel that this approach, in addition to providing a cleaner and simpler view of the problem, is more natural to work with in the context of approximation algorithms, and should be more amenable to the powerful geometric techniques in the field.

Independent Work. Independently from our work, Lecomte and Weinstein [18] showed that second Wilber Bound (WB-2) dominates the WB-1 bound, and moreover, they show an access sequence X for which the two bounds have a gap of $\Omega(\log\log n)$. In particular, their result implies that the gap between $\mathsf{WB}(X)$ and $\mathsf{OPT}(X)$ is $\Omega(\log\log n)$ for that access sequence. We note that the access sequence X that is used in our negative results also provides a gap of $\Omega(\log\log n/\log\log\log n)$ between the WB-2 and the WB-1 bounds, although we only realized this after hearing the statement of the results of [18]. Additionally, Lecomte and Weinstein show that the WB-2 bound is invariant under rotations, and use this to show that, when the WB-2 bound is constant, then the Independent Rectangle bound of [9] is linear.

We now provide a more detailed description of our results.

Our Results and Techniques

We use the geometric interpretation of the problem, introduced by Demaine et al. [9], that we refer to as the Min-Sat problem. Let P be any set of points in the plane. We say that two points $p, q \in P$ are *collinear* iff either their x-coordinates are equal, or their y-coordinates are equal. If p and q are non-collinear, then we let $\square_{p,q}$ be the smallest closed rectangle containing both p and q; notice that p and q must be diagonally opposite corners of this rectangle. We say that the pair (p, q) of points is *satisfied* in P iff there is some additional point $r \neq p, q$ in P that lies in $\square_{p,q}$ (the point may lie on the boundary of the rectangle). Lastly, we say that the set P of points is satisfied iff for every pair $p, q \in P$ of distinct points, either p and q are collinear, or they are satisfied in P.

In the Min-Sat problem, the input is a set P of points in the plane with integral x- and y-coordinates; we assume that all x-coordinates are between 1 and n, and all y-coordinates are between 1 and m and distinct from each other, and that $|P| = m$. The goal is to find a minimum-cardinality set Y of points, such that the set $P \cup Y$ of points is satisfied.

An access sequence X over keys $\{1, \ldots, n\}$ can be represented by a set P of points in the plane as follows: if a key x is accessed at time y, then add the point (x, y) to P. Demaine et al. [9] showed that, for every access sequence X, if we denote by P the corresponding set of points in the plane, then the value of the optimal solution to the Min-Sat problem on P is $\Theta(\mathsf{OPT}(X))$. They also showed that, in order to obtain an $O(\alpha)$-approximation algorithm for BST's, it is sufficient to obtain an α-approximation algorithm for the Min-Sat problem. In the online version of the Min-Sat problem, at every time step t, we discover the unique input point whose y-coordinate is t, and we need to decide which points with y-coordinate t to add to the solution. Demaine et al. [9] also showed that an α-competitive online algorithm for Min-Sat implies an $O(\alpha)$-competitive online algorithm for BST's. For convenience, we do not distinguish between the input access sequence X and the corresponding set of points in the plane, that we also denote by X.

Negative Results for WB-1. We say that an input access sequence X is a *permutation* if each key in $\{1, \ldots, n\}$ is accessed exactly once. Equivalently, in the geometric view, every column with an integral x-coordinate contains exactly one input point.

Informally, the WB-1 bound for an input sequence X is defined as follows. Let B be the bounding box containing all points of X, and consider any vertical line L drawn across B, that partitions it into two vertical strips, separating the points of X into two subsets X_1 and X_2. Assume that the points of X are ordered by their y-coordinates from smallest to largest. We say that a pair $(x, x') \in X$ of points *cross* the line L, iff x and x' are consecutive points of X, and they lie on different sides of L. Let $C(L)$ be the number of all pairs of points in X that cross L. We then continue this process recursively with X_1 and X_2, with the final value of the WB-1 bound being the sum of the two resulting bounds obtained for X_1 and X_2, and $C(L)$. This recursive partitioning process can be represented by a binary tree T that we call a *partitioning tree* (we note that the partitioning tree is not related to the BST tree that the BST algorithm maintains). Every vertex v of the partitioning tree is associated with a vertical strip $S(v)$, where for the root vertex r, $S(r) = B$. If the partitioning algorithm uses a vertical line L to partition the strip $S(v)$ into two sub-strips S_1 and S_2, then vertex v has two children, whose corresponding strips are S_1 and S_2. Note that every sequence of vertical lines used in the recursive partitioning procedure corresponds to a unique partitioning tree and vice versa. Given a set X of points and a partitioning tree T, we denote by $\mathsf{WB}_T(X)$ the WB-1 bound obtained for X while following the partitioning scheme defined by T. Wilber [25] showed that, for every partitioning tree T, $\mathsf{OPT}(X) \geq \Omega(\mathsf{WB}_T(X))$ holds. Moreover, Demaine et al. [10]

showed that, if T is a balanced tree, then $\mathsf{OPT}(X) \leq O(\log\log n) \cdot \mathsf{WB}_T(X)$. These two bounds are used to obtain the $O(\log\log n)$-competitive algorithm of [10]. We call this variant of WB-1, that is defined with respect to a fixed tree T, the *weak* WB-1 bound.

Unfortunately, it is well-known (see e.g. [10, 24, 16]), that the gap between $\mathsf{OPT}(X)$ and the weak WB-1 bound on an input X may be as large as $\Omega(\log\log n)$. In other words, for any fixed partitioning tree T, there exists an input X (that depends on T), with $\mathsf{WB}_T(X) \leq O(\mathsf{OPT}(X)/\log\log n)$. However, the construction of this "bad" input X depends on the fixed partitioning tree T. We consider a stronger variant of WB-1, that we refer to as *strong* WB-1 bound and denote by $\mathsf{WB}(X)$, that maximizes the weak WB-1 bound over all such partitioning trees, that is, $\mathsf{WB}(X) = \max_T\{\mathsf{WB}_T(X)\}$. Using this stronger bound as an alternative to weak WB-1 in order to obtain better approximation algorithms was suggested by Iacono [16], and by Kozma [17].

Our first result rules out this approach: we show that, even for the strong WB-1 bound, the gap between $\mathsf{WB}(X)$ and $\mathsf{OPT}(X)$ may be as large as $\Omega(\log\log n/\log\log\log n)$, even if the input X is a permutation.

▶ **Theorem 1.** *For every integer n', there is an integer $n \geq n'$, and an access sequence X on n keys with $|X| = n$, such that X is a permutation, $\mathsf{OPT}(X) \geq \Omega(n\log\log n)$, but $\mathsf{WB}(X) \leq O(n\log\log\log n)$. In other words, for every partitioning tree T, $\frac{\mathsf{OPT}(X)}{\mathsf{WB}_T(X)} \geq \Omega\left(\frac{\log\log n}{\log\log\log n}\right)$.*

We note that it is well known (see e.g. [5]), that any c-approximation algorithm for permutation input can be turned into an $O(c)$-approximation algorithm for any input sequence. However, the known instances that achieve an $\Omega(\log\log n)$-gap between the weak WB-1 bound and OPT are not permutations. Therefore, our result is the first to provide a super-constant gap between WB-1 and OPT for permutations, even for the case of weak WB-1.

Extension of WB-1. We consider several generalizations of the WB-1 bound that allow partitioning the plane both horizontally and vertically. We call the new bounds the *consistent Guillotine Bound* and the *Guillotine Bound*. Our negative result extends to the consistent Guillotine Bound but *not* to the Guillotine Bound. The Guillotine Bound seems to maintain the algorithm-friendly nature of WB-1, and in particular it naturally fits into the algorithmic framework that we propose. We hope that this bound can lead to improved algorithms, both in the offline and the online settings

Separating the Two Wilber Bounds. The sequence X given by Theorem 1 not only provides a separation between WB-1 and OPT, but it also provides a separation between the WB-1 bound and the WB-2 bound (also called the *funnel* bound). The latter can be defined in the geometric view as follows. Recall that, for a pair of points $x, y \in X$, $\square_{x,y}$ is the smallest closed rectangle containing both x and y. For a point x in the access sequence X, the *funnel* of x is the set of all points $y \in X$, for which $\square_{x,y}$ does not contain any point of $X \setminus \{x, y\}$, and $\mathrm{alt}(x)$ is the number of alterations between the left of x and the right of x in the funnel of x. The second Wilber Bound for sequence X is then defined as: $\mathsf{WB}^{(2)}(X) = |X| + \sum_{x\in X}\mathrm{alt}(x)$. We show that, for the sequence X given by Theorem 1, $\mathsf{WB}^{(2)}(X) \geq \Omega(n\log\log n)$ holds, and therefore $\mathsf{WB}^{(2)}(X)/\mathsf{WB}(X) \geq \Omega(\log\log n/\log\log\log n)$ for that sequence, implying that the gap between $\mathsf{WB}(X)$ and $\mathsf{WB}^{(2)}(X)$ may be as large as $\Omega(\log\log n/\log\log\log n)$. We note that we only realized that our results provide this stronger separation between the two Wilber bounds after hearing the statements of the results from the independent work of Lecomte and Weinstein [18] mentioned above.

Algorithmic Results. We provide new simple approximation algorithms for the problem, that rely on its geometric interpretation, namely the Min-Sat problem.

▶ **Theorem 2.** *There is an offline algorithm for Min-Sat, that, given any integral parameter $D \geq 1$, and an access sequence X to n keys of length m, produces a solution of cost at most $O(D \cdot OPT(X))$ and has running time at most* $\text{poly}(m) \cdot \exp\left(O\left(n^{1/2^{\Theta(D)}} \log n\right)\right)$. *For $D = O(\log \log n)$, the algorithm's running time is polynomial in n and m, and it can be adapted to the online setting, achieving an $O(\log \log n)$-competitive ratio.*

Our results show that the problem of obtaining a constant-factor approximation for Min-Sat cannot be NP-hard, unless NP $\subseteq$ SUBEXP, where SUBEXP $= \bigcap_{\epsilon>0} \text{DTime}[2^{n^\epsilon}]$. This, in turn, provides a positive evidence towards the dynamic optimality conjecture, as one natural avenue to disproving it is to show that obtaining a constant-factor approximation for BST's is NP-hard. Our results rule out this possibility, unless NP $\subseteq$ SUBEXP. While the $O(\log \log n)$-approximation factor achieved by our algorithm in time $\text{poly}(mn)$ is similar to that achieved by other known algorithms [10, 14, 24], this is the first algorithm that relies solely on the geometric formulation of the problem, which is arguably cleaner, simpler, and better suited for exploiting the rich toolkit of algorithmic techniques developed in the areas of online and approximation algorithms.

Organization. We start with preliminaries in Section 2. In Section 3, we state decomposition theorems which are useful for both of our negative and positive results. In Section 4, we provide the proof of Theorem 1, our main negative result. We discuss extensions of the Wilber Bound in Section 5. Lastly, we show our main positive result – the proof of Theorem 2 – in Section 6. Due to lack of space, many of the proofs are deferred to the full version.

2 Preliminaries

All our results only use the geometric interpretation of the problem, that we refer to as the Min-Sat problem. We include the formal definition of algorithms for BST's and formally state their equivalence to Min-Sat in the full version.

2.1 The Min-Sat Problem

For a point $p \in \mathbb{R}^2$ in the plane, we denote by $p.x$ and $p.y$ its x- and y-coordinates, respectively. Given any pair p, p' of points, we say that they are *collinear* if $p.x = p'.x$ or $p.y = p'.y$. If p and p' are not collinear, then we let $\square_{p,p'}$ be the smallest closed rectangle containing both p and p'; note that p and p' must be diagonally opposite corners of the rectangle.

▶ **Definition 3.** *We say that a non-collinear pair p, p' of points is* satisfied by a point p'' *if p'' is distinct from p and p' and $p'' \in \square_{p,p'}$ (where p'' may lie on the boundary of the rectangle). We say that a set S of points is* satisfied *iff for every non-collinear pair $p, p' \in S$ of points, there is some point $p'' \in S$ that satisfies this pair.*

We refer to horizontal and vertical lines as *rows* and *columns* respectively. For a collection of points X, the *active rows* of X are the rows that contain at least one point in X. We define the notion of *active columns* analogously. We denote by $r(X)$ and $c(X)$ the number of active rows and active columns of the point set X, respectively. We say that a point set X is a *semi-permutation* if every active row contains exactly one point of X. Note that, if X is a semi-permutation, then $c(X) \leq r(X)$. We say that X is a *permutation* if it is a

semi-permutation, and additionally, every active column contains exactly one point of X. Clearly, if X is a permutation, then $c(X) = r(X) = |X|$. We denote by B the smallest closed rectangle containing all points of X, and call B the *bounding box*.

We are now ready to define the Min-Sat problem. The input to the problem is a set X of points that is a semi-permutation, and the goal is to compute a minimum-cardinality set Y of points, such that $X \cup Y$ is satisfied. We say that a set Y of points is a *feasible solution* for X if $X \cup Y$ is satisfied. We denote by $\mathsf{OPT}(X)$ the minimum value $|Y|$ of any feasible solution Y for X.[1] In the online version of the Min-Sat problem, at every time step t, we discover the unique input point from X whose y-coordinate is t, and we need to decide which points with y-coordinate t to add to the solution Y. The Min-Sat problem is equivalent to the BST problem, in the following sense:

▶ **Theorem 4** ([9]). *Any efficient α-approximation algorithm for Min-Sat can be transformed into an efficient $O(\alpha)$-approximation algorithm for BST's, and similarly any online α-competitive algorithm for Min-Sat can be transformed into an online $O(\alpha)$-competitive algorithm for BST's.*

2.2 Basic Geometric Properties

The following observation is well known (see, e.g. Observation 2.1 from [9]).

▶ **Observation 5.** *Let Z be any satisfied point set. Then for every pair $p, q \in Z$ of distinct points, there is a point $r \in \square_{p,q} \setminus \{p, q\}$ such that $r.x = p.x$ or $r.y = p.y$.*

Collapsing Sets of Columns or Rows. Assume that we are given any set X of points, and any collection $\mathcal{C}$ of consecutive active columns for X. In order to collapse the set $\mathcal{C}$ of columns, we replace $\mathcal{C}$ with a single representative column C (for concreteness, we use the column of $\mathcal{C}$ with minimum x-coordinate). For every point $p \in X$ that lies on a column of $\mathcal{C}$, we replace p with a new point, lying on the column C, whose y-coordinate remains the same. Formally, we replace point p with point $(x, p.y)$, where x is the x-coordinate of the column C. We denote by $X_{|\mathcal{C}}$ the resulting new set of points. We can similarly define collapsing set of rows. The following useful observation is easy to verify.

▶ **Observation 6.** *Let S be any set of points, and let $\mathcal{C}$ be any collection of consecutive active columns (or rows) with respect to S. If S is a satisfied set of points, then so is $S_{|\mathcal{C}}$.*

Canonical Solutions. We say that a solution Y for input X is *canonical* iff every point $p \in Y$ lies on an active row and an active column of X. It is easy to see that *any* solution can be transformed into a canonical solution, without increasing its cost (see the full version of the paper for the proof).

▶ **Observation 7.** *There is an efficient algorithm, that, given an instance X of Min-Sat and any feasible solution Y for X, computes a feasible canonical solution $\hat{Y}$ for X with $|\hat{Y}| \leq |Y|$.*

2.3 Partitioning Trees

We now turn to define partitioning trees, that are central to both defining the WB-1 bound and to describing our algorithm.

[1] We remark that in the original paper that introduced this problem [9], the value of the solution is defined as $|X \cup Y|$, while our solution value is $|Y|$. It is easy to see that for any semi-permutation X and solution Y for X, $|Y| \geq \Omega(|X|)$ must hold, so the two definitions are equivalent to within factor 2.

Let X be the a set of points that is a semi-permutation. We can assume without loss of generality that every column with an integral x-coordinate between 1 and $c(X)$ inclusive contains at least one point of X. Let B be the bounding box of X. Assume that the set of active columns is $\{C_1, \ldots, C_a\}$, where $a = c(X)$, and that for all $1 \leq i \leq a$, the x-coordinate of column C_i is i. Let $\mathcal{L}$ be the set of all vertical lines with half-integral x-coordinates between $1 + 1/2$ and $a - 1/2$ (inclusive). Throughout, we refer to the vertical lines in $\mathcal{L}$ as *auxiliary columns*. Let σ be an arbitrary ordering of the lines of $\mathcal{L}$ and denote $\sigma = (L_1, L_2, \ldots, L_{a-1})$. We define a hierarchical partition of the bounding box B into vertical strips using σ, as follows. We perform $a - 1$ iterations. In the first iteration, we partition the bounding box B, using the line L_1, into two vertical strips, S_L and S_B. For $1 < i \leq a - 1$, in iteration i we consider the line L_i, and we let S be the unique vertical strip in the current partition that contains the line L_i. We then partition S into two vertical sub-strips by the line L_i. When the partitioning algorithm terminates, every vertical strip contains exactly one active column.

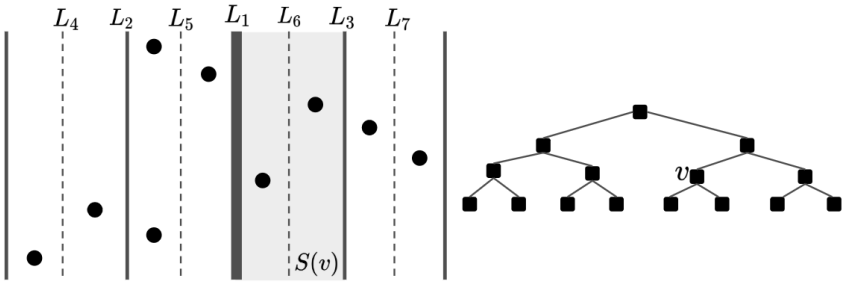


Figure 1 An Illustration of partitioning tree and the corresponding sequence $\sigma = (L_1, \ldots, L_7)$. Strip $S(v)$ corresponds to node v that owns line L_6.

This partitioning process can be naturally described by a binary tree $T = T(\sigma)$, that we call a *partitioning tree* associated with the ordering σ (see Figure 1). Each node $v \in V(T)$ is associated with a vertical strip $S(v)$ of the bounding box B. The strip $S(r)$ of the root vertex r of T is the bounding box B. For every inner vertex $v \in V(T)$, if $S = S(v)$ is the vertical strip associated with v, and if $L \in \mathcal{L}$ is the first line in σ that lies strictly in S, then line L partitions S into two sub-strips, that we denote by S_L and S_R. Vertex v then has two children, whose corresponding strips are S_L and S_R respectively. We say that v *owns* the line L, and we denote $L = L(v)$. For each leaf node v, the corresponding strip $S(v)$ contains exactly one active column of X, and v does not own any line of $\mathcal{L}$. For each vertex $v \in V(T)$, let $N(v) = |X \cap S(v)|$ be the number of points from X that lie in $S(v)$, and let width(v) be the width of the strip $S(v)$. Given a partition tree T for point set X, we refer to the vertical strips in $\{S(v)\}_{v \in T}$ as T-strips.

2.4 The WB-1 Bound

The WB-1 bound[2] is defined with respect to an ordering (or a permutation) σ of the auxiliary columns, or, equivalently, with respect to the partitioning tree $T(\sigma)$. It will be helpful to keep both these views in mind. In this paper, we will make a clear distinction between a weak variant of the WB-1 bound, as defined by Wilber himself in [25] and a strong variant, as mentioned in [16].

[2] Also called Interleaving bound [10], the first Wilber bound, "interleave lower bound" [25], or alternation bound [16]

Let X be a semi-permutation, and let $\mathcal{L}$ be the corresponding set of auxiliary columns. Consider an arbitrary fixed ordering σ of columns in $\mathcal{L}$ and its corresponding partition tree $T = T(\sigma)$. For each inner node $v \in V(T)$, consider the set $X' = X \cap S(v)$ of input points that lie in the strip $S(v)$, and let $L(v) \in \mathcal{L}$ be the line that v owns. We denote $X' = \{p_1, p_2, \ldots, p_k\}$, where the points are indexed in the increasing order of their y-coordinates; since X is a semi-permutation, no two points of X may have the same y-coordinate. For $1 \leq j < k$, we say that the ordered pair (p_j, p_{j+1}) of points form a *crossing* of $L(v)$ iff p_j, p_{j+1} lie on the opposite sides of the line $L(v)$. We let $\mathsf{cost}(v)$ be the total number of crossings of $L(v)$ by the points of $X \cap S(v)$. When $L = L(v)$, we also write $\mathsf{cost}(L)$ to denote $\mathsf{cost}(v)$. If v is a leaf vertex, then its cost is set to 0.

▶ **Definition 8** (WB-1 bound). *For any semi-permutation X, an ordering σ of the auxiliary columns in $\mathcal{L}$, and the corresponding partitioning tree $T = T_\sigma$, the (weak) WB-1 bound of X with respect to σ is: $\mathsf{WB}_\sigma(X) = \mathsf{WB}_T(X) = \sum_{v \in V(T)} \mathsf{cost}(v)$. The strong WB-1 bound of X is $\mathsf{WB}(X) = \max_\sigma \mathsf{WB}_\sigma(X)$, where the maximum is taken over all permutations σ of the lines in $\mathcal{L}$.*

It is well known that the WB-1 bound is a lower bound on the optimal solution cost:

▷ Claim 9. For any semi-permutation X, $\mathsf{WB}(X) \leq 2 \cdot \mathsf{OPT}(X)$.

The original proof of this fact is due to Wilber [25], which was later presented in the geometric view by Demaine et al. [9], via the notion of *independent rectangles*.

3 Geometric Decomposition Theorems

In this section, we develop several technical tools that will allow us to decompose a given instance into a number of sub-instances. We then analyze the optimal solution costs and the Wilber bound values for the resulting subinstances.

Split Instances. Consider a semi-permutation X and its partitioning tree T. Let $U \subseteq V(T)$ be a collection of vertices of the tree T, such that the strips $\{S(v)\}_{v \in U}$ partition the bounding box. In other words, every root-to-leaf path in T must contain exactly one vertex of U. We now define splitting an instance X via the set U of vertices of T.

▶ **Definition 10** (A Split). *A* split *of (X, T) at U is a collection of instances $\{X^c, \{X^s_v\}_{v \in U}\}$, defined as follows.*

- *For each vertex $v \in U$, instance X^s_v is called a* ***strip instance****, and it contains all points of X that lie in the interior of the strip $S(v)$.*
- *Instance X^c is called a* ***compressed instance****, and it is obtained from X by collapsing, for every vertex $v \in U$, all active columns in the strip $S(v)$ into a single column.*

We also partition the tree T into sub-trees that correspond to the new instances: for every vertex $v \in U$, we let T_v be the sub-tree of T rooted at v. Observe that T_v is a partitioning tree for instance X^s_v. The tree T^c is obtained from T by deleting from it, for all $v \in U$, all vertices of $V(T_v) \setminus \{v\}$. It is easy to verify that T^c is a valid partitioning tree for instance X^c. The following observation, whose proof appears in the full version of the paper, establishes several basic properties of a split. Recall that, given an instance X, $r(X)$ and $c(X)$ denote the number of active rows and active columns in X, respectively.

▶ **Observation 11.** *If X is a semi-permutation, then the following properties hold for any (X,T)-split at U:*

- $\sum_{v\in U} r(X^s_v) = r(X)$
- $\sum_{v\in U} c(X^s_v) = c(X)$
- $c(X^c) \leq |U|$
- $\sum_{v\in U} \mathsf{WB}_{T_v}(X^s_v) + \mathsf{WB}_{T^c}(X^c) = \mathsf{WB}_T(X)$.

The first property holds since X is a semi-permutation. In order to establish the last property, consider any vertex $x \in V(T)$, and let $T' \in \{T^c\} \cup \{T_v\}_{v\in U}$ be the new tree to which v belongs; if $x \in U$, then we set $T' = T_x$. It is easy to see that the cost of v in tree T' is the same as its cost in the tree T (recall that the cost of a leaf vertex is 0). The last property can be viewed as a "perfect decomposition" property of the weak WB-1 bound. We will show below an (approximate) decomposition property of strong WB-1 bound.

Splitting by Lines. We can also define the splitting with respect to any subset $\mathcal{L}' \subseteq \mathcal{L}$ of the auxiliary columns for X, analogously: Notice that the lines in $\mathcal{L}'$ partition the bounding box B into a collection of internally disjoint strips, that we denote by $\{S'_1, \ldots, S'_k\}$. We can then define the strip instances X^s_i as containing all vertices of $X \cap S_i$ for all $1 \leq i \leq k$, and the compressed instance X^c, that is obtained by collapsing, for each $1 \leq i \leq k$, all active columns that lie in strip S_i, into a single column. We also call these resulting instances *a split of X by $\mathcal{L}'$*.

We can also consider an arbitrary ordering σ of the lines in $\mathcal{L}$, such that the lines of $\mathcal{L}'$ appear at the beginning of σ, and let $U \subseteq V(T(\sigma))$ contain all vertices u for which the strip $S(u)$ is in $\{S_i\}_{1\leq i\leq k}$. If we perform a split of (X,T) at U, we obtain exactly the same strip instances $X^s_1, \ldots, X^s_k$, and the same compressed instance X^c.

Decomposition Theorem for OPT. The following theorem gives a crucial decomposition property of OPT. The theorem is used in our algorithm for Min-Sat, and its proof appears in the full version of the paper.

▶ **Theorem 12.** *Let X be a semi-permutation, let T be any partitioning tree for X, let $U \subseteq V(T)$ be a subset of vertices of T such that the strips in $\{S(v) \mid v \in U\}$ partition the bounding box, and let $\{X^c, \{X^s_v\}_{v\in U}\}$ be an (X,T)-split at U. Then:*

$$\sum_{v\in U} \mathit{OPT}(X^s_v) + \mathit{OPT}(X^c) \leq \mathit{OPT}(X).$$

Decomposition Theorem for the Strong WB-1 bound. We also prove, in the full version of the paper, the following theorem about the strong WB-1 bound, that we use several times in our negative result.

▶ **Theorem 13.** *Let X be a semi-permutation and let T be a partitioning tree for X. Let $U \subseteq V(T)$ be a set of vertices of T such that the strips in $\{S(v) \mid v \in U\}$ partition the bounding box. Let $\{X^c, \{X^s_v\}_{v\in U}\}$ be the split of (X,T) at U. Then:*

$$\mathsf{WB}(X) \leq 4\mathsf{WB}(X^c) + 8\sum_{v\in U} \mathsf{WB}(X^s_v) + O(|X|).$$

This result is somewhat surprising. One can think of the expression $\mathsf{WB}(X^c) + \sum_{v\in U} \mathsf{WB}(X^s_v)$ as a WB-1 bound obtained by first cutting along the lines that serve as boundaries of the strips $S(v)$ for $v \in U$, and then cutting the individual strips. However,

$\mathsf{WB}(X)$ is the maximum of $\mathsf{WB}_T(X)$ obtained over all trees T, including those that do not obey this partitioning order. The proofs of both Theorems 12 and 13 are given in the full version.

4 Separation of OPT and the Strong Wilber Bound

In this section we present our negative results, proving Theorem 1. We start by defining several basic tools used in our construction in Section 4.1. From Section 4.2 onward, we describe our construction and its analysis.

4.1 Basic Tools

Monotonically Increasing Sequence. We say that an input set X of points is *monotonically increasing* iff X is a permutation, and moreover for every pair $p, p' \in X$ of points, if $p.x < p'.x$, then $p.y < p'.y$ must hold. It is well known that the value of the optimal solution of monotonically increasing sequences is low, and we exploit this fact in our negative results.

▶ **Observation 14.** *If X is a monotonically increasing set of points, then $\mathsf{OPT}(X) \leq |X| - 1$.*

Bit Reversal Sequence (BRS). We use the geometric variant of BRS, which is more intuitive and easier to argue about. Let $\mathcal{R} \subseteq \mathbb{N}$ and $\mathcal{C} \subseteq \mathbb{N}$ be sets of integers (which are supposed to represent sets of active rows and columns.) The instance $\mathsf{BRS}(i, \mathcal{R}, \mathcal{C})$ is only defined when $|\mathcal{R}| = |\mathcal{C}| = 2^i$. It contains 2^i points, and it is a permutation, whose sets of active rows and columns are exactly $\mathcal{R}$ and $\mathcal{C}$ respectively; so $|\mathcal{R}| = |\mathcal{C}| = 2^i$. We define the instance recursively. The base of the recursion is instance $\mathsf{BRS}(0, \{C\}, \{R\})$, containing a single point at the intersection of row R and column C. Assume now that we have defined, for all $1 \leq i' \leq i$, and any sets $\mathcal{R}', \mathcal{C}'$ of $2^{i'}$ integers, the corresponding instance $\mathsf{BRS}(i, \mathcal{R}', \mathcal{C}')$. We define instance $\mathsf{BRS}(i+1, \mathcal{R}, \mathcal{C})$, where $|\mathcal{R}| = |\mathcal{C}| = 2^{i+1}$, as follows.

Consider the columns in $\mathcal{C}$ in their natural left-to-right order, and define $\mathcal{C}_{left}$ to be the first 2^i columns and $\mathcal{C}_{right} = \mathcal{C} \setminus \mathcal{C}_{left}$. Denote $\mathcal{R} = \{R_1, \ldots, R_{2^{i+1}}\}$, where the rows are indexed in their natural bottom to top order, and let $\mathcal{R}_{even} = \{R_2, R_4, \ldots, R_{2^{i+1}}\}$ and $\mathcal{R}_{odd} = \{R_1, R_3, \ldots, R_{2^{i+1}-1}\}$ be the sets of all even-indexed and all odd-indexed rows, respectively. Notice that $|\mathcal{C}_{left}| = |\mathcal{C}_{right}| = |\mathcal{R}_{even}| = |\mathcal{R}_{odd}| = 2^i$. The instance $\mathsf{BRS}(i+1, \mathcal{R}, \mathcal{C})$ is defined to be $\mathsf{BRS}(i, \mathcal{R}_{odd}, \mathcal{C}_{left}) \cup \mathsf{BRS}(i, \mathcal{R}_{even}, \mathcal{C}_{right})$.

For $n = 2^i$, we denote by $\mathsf{BRS}(n)$ the instance $\mathsf{BRS}(i, \mathcal{C}, \mathcal{R})$, where $\mathcal{C}$ contains all columns with integral x-coordinates from 1 to n, and $\mathcal{R}$ contains all rows with integral y-coordinates from 1 to n; see Figure 2 for an illustration.

It is well-known that, if X is a bit-reversal sequence on n points, then $\mathsf{OPT}(X) \geq \Omega(n \log n)$.

▷ Claim 15. Let $X = \mathsf{BRS}(i, \mathcal{C}, \mathcal{R})$, for any $i \geq 0$ and any sets $\mathcal{C}$ and $\mathcal{R}$ of columns and rows, respectively, with $|\mathcal{R}| = |\mathcal{C}| = 2^i$. Then $|X| = 2^i$, and $\mathsf{OPT}(X) \geq \frac{\mathsf{WB}(X)}{2} \geq \frac{|X|(\log |X|-2)+1}{2}$.

Next, we present two additional technical tools that we use in our construction.

Exponentially Spaced Columns. Recall that we defined the bit reversal instance $\mathsf{BRS}(\ell, \mathcal{R}, \mathcal{C})$, where $\mathcal{R}$ and $\mathcal{C}$ are sets of 2^ℓ rows and columns, respectively, that serve in the resulting instance as the sets of active rows and columns; the instance contains $n = 2^\ell$ points. In the Exponentially-Spaced BRS instance $\mathsf{ES\text{-}BRS}(\ell, \mathcal{R})$, we are still given a set $\mathcal{R}$ of 2^ℓ rows that will serve as active rows in the resulting instance, but we define the set

$\mathcal{C}$ of columns in a specific way. For an integer i, C_i be the column whose x-coordinate is i. We then let $\mathcal{C}$ contain, for each $0 \leq i < 2^\ell$, the column C_{2^i}. Denoting $N = 2^n = 2^{2^\ell}$, the x-coordinates of the columns in $\mathcal{C}$ are $\{1, 2, 4, 8, \ldots, N/2\}$. The instance is then defined to be $\mathsf{BRS}(\ell, \mathcal{R}, \mathcal{C})$ for this specific set $\mathcal{C}$ of columns. Notice that the instance contains $n = \log N = 2^\ell$ input points.

It is easy to see that any point set $X = \mathsf{ES\text{-}BRS}(\ell, \mathcal{R})$ satisfies $\mathsf{OPT}(X) = \Omega(n \log n)$. We remark that this idea of exponentially spaced columns is inspired by the instance used by Iacono [16] to prove a gap between the weak WB-1 bound and $\mathsf{OPT}(X)$. However, Iacono's instance is tailored to specific partitioning tree T, and it is clear that there is another partitioning tree T' with $\mathsf{OPT}(X) = \Theta(WB_{T'}(X))$. Therefore, this instance does not give a separation result for the strong WB-1 bound, and in fact it does not provide negative results for the weak WB-1 bound when the input point set is a permutation.

Cyclic Shift of Columns. Suppose we are given a point set X, and let $\mathcal{C}' = \{C_0, \ldots, C_{N-1}\}$ be any set of columns indexed in their natural left-to-right order, such that all points of X lie on columns of $\mathcal{C}$ (but some columns may contain no points of X). Let $0 \leq s < N$ be any integer. We denote by X^s a cyclic shift of X by s units, obtained as follows. For every point $p \in X$, we add a new point p^s to X^s, whose y-coordinate is the same as that of p, and whose x-coordinate is $p.x + s \mod N$. In other words, we shift the point p by s steps to the right in a circular manner. Equivalently, we move the last s columns of $\mathcal{C}'$ to the beginning of the instance. The following claim, whose proof appears in the full version of the paper, shows that the value of the optimal solution does not decrease significantly in the shifted instance.

▷ Claim 16. Let X be any point set that is a semi-permutation. Let $0 \leq s < N$ be a shift value, and let $X' = X^s$ be the instance obtained from X by a cyclic shift of its points by s units to the right. Then $\mathsf{OPT}(X') \geq \mathsf{OPT}(X) - |X|$.

4.2 Construction of the Bad Instance

We construct two instances: instance $\hat{X}$ on N^* points, that is a semi-permutation (but is somewhat easier to analyze), and instance X^* in N^* points, which is a permutation; the analysis of instance X^* heavily relies on the analysis of instance $\hat{X}$. We will show that the optimal solution value of both instances is $\Omega(N^* \log\log N^*)$, but the cost of the Wilber Bound is at most $O(N^* \log\log\log N^*)$. Our construction uses the following three parameters. We let $\ell \geq 1$ be an integer, and we set $n = 2^\ell$ and $N = 2^n$.

First Instance. We now construct our first final instance $\hat{X}$, which is a semi-permutation containing N columns. Intuitively, we create N instances $X^0, X^1, \ldots, X^{N-1}$, where instance X^s is an exponentially-spaced BRS instance that is shifted by s units. We then stack these instances on top of one another in this order.

Formally, for all $0 \leq j \leq N-1$, we define a set $\mathcal{R}_j$ of n consecutive rows with integral coordinates, such that the rows of $\mathcal{R}_0, \mathcal{R}_1, \ldots, \mathcal{R}_{N-1}$ appear in this bottom-to-top order. Specifically, set $\mathcal{R}_j$ contains all rows whose y-coordinates are in $\{jn+1, jn+2, \ldots, (j+1)n\}$.

For every integer $0 \leq s \leq N-1$, we define a set of points X^s, which is a cyclic shift of instance $\mathsf{ES\text{-}BRS}(\ell, \mathcal{R}_s)$ by s units. Recall that $|X^s| = 2^\ell = n$ and that the points in X^s appear on the rows in $\mathcal{R}_s$ and a set $\mathcal{C}_s$ of columns, whose x-coordinates are in $\{\left(2^j + s\right) \mod N : 0 \leq j < n\}$. We then let our final instance be $\hat{X} = \bigcup_{s=0}^{N-1} X^s$. From now on, we denote $N^* = |\hat{X}|$. Recall that $|N^*| = N \cdot n = N \log N$.

Observe that the number of active columns in $\hat{X}$ is N. Since the instance is symmetric and contains $N^* = N \log N$ points, every column contains exactly $\log N$ points. Each row contains exactly one point, so $\hat{X}$ is a semi-permutation. (See Figure 2 for an illustration).

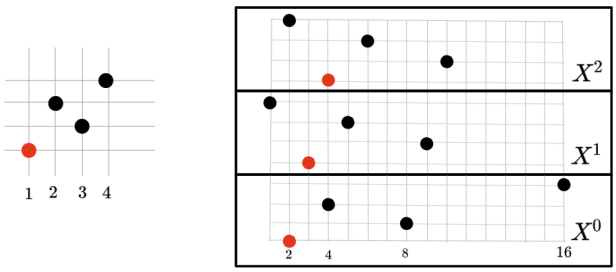


Figure 2 An illustration of our construction. The figure on the left shows the instance $\mathsf{BRS}(2, \{1,2,3,4\}, \{1,2,3,4\})$. The figure on the right combines three copies X^0, X^1, X^2 of the corresponding exponentially-spaced instance, with horizontal shifts of 0, 1, and 2, respectively. The red points show how copies of the same point in different sub-instances.

Lastly, we need the following bound on the value of the optimal solution of instance $\hat{X}$.

▶ **Observation 17.** $OPT(\hat{X}) = \Omega(N^* \log\log N^*)$

Proof. From Claims 15 and 16, for each $0 \leq s \leq N-1$, each sub-instance X^s has $\mathsf{OPT}(X^s) \geq \Omega(n \log n) = \Omega(\log N \log\log N)$. Therefore, $\mathsf{OPT}(\hat{X}) \geq \sum_{s=0}^{N-1} \mathsf{OPT}(X^s) = \Omega(N \log N \log\log N) = \Omega(N^* \log\log N^*)$ (we have used the fact that $N^* = N \log N$). ◀

Second Instance. We now construct our second and final instance, X^*, that is a permutation. In order to do so, we start with the instance $\hat{X}$, and, for every active column C of $\hat{X}$, we create $n = \log N$ new columns (that we view as copies of C), $C^1, \ldots, C^{\log N}$, which replace the column C. We denote this set of columns by $\mathcal{B}(C)$, and we refer it as the *block of columns representing* C. Recall that the original column C contains $\log N$ input points of $\hat{X}$. We place each such input point on a distinct column of $\mathcal{B}(C)$, so that the points form a monotonically increasing sequence (see the definition in Section 4.1). This completes the definition of the final instance X^*. We obtain the following immediate bound on the optimal solution cost of instance X^*.

▷ Claim 18. $\mathsf{OPT}(X^*) \geq \mathsf{OPT}(\hat{X}) = \Omega(N^* \log\log N^*)$.

4.3 Upper Bound for $\mathsf{WB}(\hat{X})$

In this section we prove the following theorem.

▶ **Theorem 19.** $\mathsf{WB}(\hat{X}) \leq O(N^* \log\log\log N^*)$.

In order to prove the theorem, consider again the instance $\hat{X}$. Recall that it consists of N instances $X^0, X^1, \ldots, X^{N-1}$ that are stacked on top of each other vertically in this order. We rename these instances as $X_1, X_2, \ldots, X_N$, so X_j is exactly $\mathsf{ES\text{-}BRS}(\log N)$, that is shifted by $(j-1)$ units to the right. Recall that $|\hat{X}| = N^* = N \log N$, and each instance X_s contains exactly $\log N$ points. We denote by $\mathcal{C}$ the set of N columns, whose x-coordinates are $1, 2, \ldots, N$. All points of $\hat{X}$ lie on the columns of $\mathcal{C}$. For convenience, for $1 \leq j \leq N$, we denote by C_j the column of $\mathcal{C}$ whose x-coordinate is j.

Let σ be any ordering of the auxiliary columns in $\mathcal{L}$, and let $T = T_\sigma$ be the corresponding partitioning tree. It is enough to show that, for any such ordering σ, the value of $\mathsf{WB}_\sigma(\hat{X})$ is bounded by $O(N^* \log\log\log N^*)$. Recall that $WB_\sigma(\hat{X})$ is the sum, over all vertices $v \in V(T)$, of $\mathsf{cost}(v)$. The value of $\mathsf{cost}(v)$ is defined as follows. If v is a leaf vertex, then $\mathsf{cost}(v) = 0$. Otherwise, let $L = L(v)$ be the line of $\mathcal{L}$ that v owns. Index the points in $X \cap S(v)$ by $q_1, \ldots, q_z$ in their bottom-to-top order. A consecutive pair (q_j, q_{j+1}) of points is a *crossing* iff they lie on different sides of $L(v)$. We distinguish between the two types of crossings that contribute towards $\mathsf{cost}(v)$. We say that the crossing (q_j, q_{j+1}) is of *type*-1 if both q_j and q_{j+1} belong to the same shifted instance X_s for some $0 \leq s \leq N-1$. Otherwise, they are of type-2. Note that, if (q_j, q_{j+1}) is a crossing of type 2, with $q_j \in X_s$ and $q_{j+1} \in X_{s'}$, then s, s' are not necessarily consecutive integers, as it is possible that for some indices s'', $X_{s''}$ has no points that lie in the strip $S(v)$. We now let $\mathsf{cost}_1(v)$ be the total number of type-1 crossings of $L(v)$, and $\mathsf{cost}_2(v)$ the total number of type-2 crossings. Note that $\mathsf{cost}(v) = \mathsf{cost}_1(v) + \mathsf{cost}_2(v)$. We also define $\mathsf{cost}_1(\sigma) = \sum_{v \in V(T)} \mathsf{cost}_1(v)$ and $\mathsf{cost}_2(\sigma) = \sum_{v \in V(T)} \mathsf{cost}_2(v)$. Clearly, $\mathsf{WB}_\sigma(\hat{X}) = \mathsf{cost}_1(\sigma) + \mathsf{cost}_2(\sigma)$. In the full version of the paper, we prove the following two theorems:

▶ **Theorem 20.** *For every ordering σ of the auxiliary columns in $\mathcal{L}$, $\mathsf{cost}_1(\sigma) \leq O(N^* \log\log\log N^*)$.*

▶ **Theorem 21.** *For every vertex $v \in V(T)$, $\mathsf{cost}_2(v) \leq O(\log N) + O(\mathsf{cost}_1(v))$.*

Notice that from the latter theorem, we get that $\mathsf{cost}_2(\sigma) \leq O(\mathsf{cost}_1(\sigma)) + O(|V(T)| \cdot \log N) = O(N^* \log\log\log N^*) + O(N \log N) = O(N^* \log\log\log N^*)$. Combining the two theorems together completes the proof of Theorem 19.

4.4 Upper Bound for $\mathsf{WB}(X^*)$

In this section we show that $\mathsf{WB}(X^*) = O(N^* \log\log\log N^*)$, completing the proof of Theorem 1. Recall that instance X^* is obtained from instance $\hat{X}$ by replacing every active column C of X^* with a block $\mathcal{B}(C)$ of columns, and then placing the points of C on the columns of $\mathcal{B}(C)$ so that they form a monotone increasing sequence, while preserving their y-coordinates. The resulting collection of all blocks $\mathcal{B}(C)$ partitions the set of all active columns of X^*. We denote this set of blocks by $\mathcal{B}_1, \ldots, \mathcal{B}_N$. The idea is to use Theorem 13 in order to bound $\mathsf{WB}(X^*)$.

Consider a set of lines $\mathcal{L}'$ (with half-integral x-coordinates) that partition the bounding box B into strips, where the ith strip contains the block $\mathcal{B}_i$ of columns, so $|\mathcal{L}'| = (N-1)$. We consider a split of instance X^* by $\mathcal{L}'$: This gives us a collection of strip instances $\{X_i^s\}_{1 \leq i \leq N}$ and the compressed instance X^c. Notice that the compressed instance is precisely $\hat{X}$, and each strip instance X_i^s is a monotone increasing point set.

Since each strip instance X_i^s is monotonously increasing, from Observation 14 and Claim 9, for all i, $\mathsf{WB}(X_i^s) \leq O(\mathsf{OPT}(X_i^s)) \leq O(|X_i^s|)$. From Theorem 13, we then get that: $\mathsf{WB}(X^*) \leq 4\mathsf{WB}(\hat{X}) + 8\sum_i \mathsf{WB}(X_i^s) + O(|X^*|) \leq 4\mathsf{WB}(\hat{X}) + O(|X^*|) \leq O(N^* \log\log\log N^*)$.

5 Guillotine Bounds

In this section we consider an extension of the Wilber bound which we call the Guillotine bound. The *Guillotine bound* $\mathsf{GB}(X)$ extends $\mathsf{WB}(X)$ by allowing both vertical and horizontal partitioning lines. Specifically, given the bounding box B, we let L be any vertical or horizontal line crossing B, that separates X into two subsets X_1 and X_2. We define the

number of crossings of L exactly as before, and then recurse on both sides of L as before. This partitioning scheme can be represented by a binary tree T, where every vertex of the tree is associated with a rectangular region of the plane. We denote the resulting bound obtained by using the partitioning tree T by $\mathsf{GB}_T(X)$, and we define $\mathsf{GB}(X) = \max_T \mathsf{GB}_T(X)$. We show that GB is a lower bound on the optimal solution cost in the following lemma, whose proof is deferred to the full version.

▶ **Lemma 22.** *For any point set X that is a permutation, $\mathsf{GB}(X) \leq 2\mathit{OPT}(X)$.*

The *Consistent Guillotine bound* restricts the Guillotine bound by maximizing only over partitioning schemes that are "consistent" in the following sense: suppose that the current partition of the bounding box B, that we have obtained using previous partitioning lines, is a collection $\{R_1, \ldots, R_k\}$ of rectangular regions. We need to choose a vertical or a horizontal line L that spans the whole bounding box B, that is, L intersects the boundary of B in two points. Once line L is chosen, for **every** rectangular region R_i that intersects L, we must partition R_i into two sub-regions using the line L, and then count the number of consecutive pairs of points in $X \cap R_i$ that cross the line L. In other words, we must partition all rectangles $R_1, \ldots, R_k$ consistently with respect to the line L. In contrast, in the Guillotine bound, we are allowed to partition each area R_i independently. From the definitions, the value of the Guillotine bound $\mathsf{GB}(X)$ is always at least as large as the value of the Consistent Guillotine bound, denoted by $\mathsf{cGB}(X)$, on any input sequence X, which is at least as large as $\mathsf{WB}(X)$. We generalize our negative result to the Consistent Guillotine bound in the following theorem, whose proof appears in the full version of the paper.

▶ **Theorem 23.** *For every integer n', there is an integer $n \geq n'$, and a set X of points that is a permutation with $|X| = n$, such that $\mathit{OPT}(X) \geq \Omega(n \log\log n)$ but $\mathsf{cGB}(X) \leq O(n \log\log\log n)$.*

Our negative results do not extend to the general GB bound, while our divide-and-conquer framework can naturally be adapted to work with GB. We leave open an interesting question of establishing the worst-case gap between the value of OPT and that of the Guillotine bound, and we hope that combining the Guillotine bound with our algorithmic framework will lead to better online and offline approximation algorithms.

6 The Algorithms

In this section we provide the high level intuition for the proof of Theorem 2. A more detailed description appears in the Appendix. Both the polynomial time and the sub-exponential time algorithms follow the same framework. We start with a high-level overview of this framework. For simplicity, assume that the number of active columns in the input instance X is an integral power of 2. The key idea is to decompose the input instance into smaller sub-instances, using the split instances defined in Section 3. We solve the resulting instances recursively and then combine the resulting solutions.

Suppose we are given an input point set X that is a semi-permutation, with $|X| = m$, such that the number of active columns is n. We consider a *balanced* partitioning tree T, where for every vertex $v \in V(T)$, the line $L(v)$ that v owns splits the strip $S(v)$ in the middle, with respect to the active columns that are contained in $S(v)$. Therefore, the height of the partitioning tree is $\log n$.

Consider now the set U of vertices of T that lie in the middle layer of T. We consider the split of (X, T) at U, obtaining a new collection of instances $(X^c, \{X^s_i\})_{i=1}^k$ where $k = \Theta(\sqrt{n})$. Note that each resulting strip instance X^s_i contains $\Theta(\sqrt{n})$ active columns, and so does the compressed instance X^c.

We recursively solve each such instance and then combine the resulting solutions. The key to the algorithm and its analysis is to show that there is a collection Z of $O(|X|)$ points, such that, if we are given any solution Y^c to instance X^c, and, for all $1 \leq i \leq k$, any solution Y_i to instance X_i^s, then $Z \cup Y^c \cup \left(\bigcup_{i=1}^{N} Y_i\right)$ is a feasible solution to instance X. We also show that the total number of input points that appear in all instances that participate in the same recursive level is bounded by $O(\mathsf{OPT}(X))$. This ensures that in every recursive level we add at most $O(\mathsf{OPT}(X))$ points to the solution, and the total solution cost is at most $O(\mathsf{OPT}(X))$ times the number of the recursive levels, which is bounded by $O(\log \log n)$.

In order to obtain the sub-exponential time algorithm, we restrict the recursion to D levels, and then solve each resulting instance directly in time $r(X)c(X)^{O(c(X))}$. This approach gives an $O(D)$-approximation algorithm with running time at most $\mathrm{poly}(m) \cdot \exp\left(n^{1/2^{\Omega(D)}} \log n\right)$ as desired. A more detailed description of the algorithm appears in the Appendix.

References

1. G. M. Adelson-Velskiĭ and E. M. Landis. An algorithm for organization of information. *Dokl. Akad. Nauk SSSR*, 146:263–266, 1962.
2. Rudolf Bayer. Symmetric binary b-trees: Data structure and maintenance algorithms. *Acta Inf.*, 1:290–306, 1972. doi:10.1007/BF00289509.
3. Prosenjit Bose, Karim Douïeb, John Iacono, and Stefan Langerman. The power and limitations of static binary search trees with lazy finger. In *Algorithms and Computation - 25th International Symposium, ISAAC 2014, Jeonju, Korea, December 15-17, 2014, Proceedings*, pages 181–192, 2014. doi:10.1007/978-3-319-13075-0_15.
4. P. Chalermsook, M. Goswami, L. Kozma, K. Mehlhorn, and T. Saranurak. Greedy is an almost optimal deque. *WADS*, 2015.
5. Parinya Chalermsook, Mayank Goswami, László Kozma, Kurt Mehlhorn, and Thatchaphol Saranurak. Pattern-avoiding access in binary search trees. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 410–423, 2015.
6. R. Chaudhuri and H. Höft. Splaying a search tree in preorder takes linear time. *SIGACT News*, 24(2):88–93, April 1993. doi:10.1145/156063.156067.
7. R. Cole. On the dynamic finger conjecture for splay trees. part ii: The proof. *SIAM Journal on Computing*, 30(1):44–85, 2000. doi:10.1137/S009753979732699X.
8. Richard Cole, Bud Mishra, Jeanette Schmidt, and Alan Siegel. On the dynamic finger conjecture for splay trees. part i: Splay sorting log n-block sequences. *SIAM J. Comput.*, 30(1):1–43, April 2000. doi:10.1137/S0097539797326988.
9. Erik D. Demaine, Dion Harmon, John Iacono, Daniel M. Kane, and Mihai Pătraşcu. The geometry of binary search trees. In *SODA 2009*, pages 496–505, 2009. URL: http://dl.acm.org/citation.cfm?id=1496770.1496825.
10. Erik D. Demaine, Dion Harmon, John Iacono, and Mihai Pătraşcu. Dynamic optimality - almost. *SIAM J. Comput.*, 37(1):240–251, 2007. Announced at FOCS'04. doi:10.1137/S0097539705447347.
11. Jonathan Derryberry, Daniel Dominic Sleator, and Chengwen Chris Wang. A lower bound framework for binary search trees with rotations, 2005.
12. Jonathan C. Derryberry and Daniel D. Sleator. Skip-splay: Toward achieving the unified bound in the BST model. In Frank Dehne, Marina Gavrilova, Jörg-Rüdiger Sack, and CsabaD. Toth, editors, *Algorithms and Data Structures*, volume 5664 of *Lecture Notes in Computer Science*, pages 194–205. Springer Berlin Heidelberg, 2009. doi:10.1007/978-3-642-03367-4_18.
13. Amr Elmasry. On the sequential access theorem and deque conjecture for splay trees. *Theoretical Computer Science*, 314(3):459–466, 2004. doi:10.1016/j.tcs.2004.01.019.

14 George F. Georgakopoulos. Chain-splay trees, or, how to achieve and prove loglogn-competitiveness by splaying. *Inf. Process. Lett.*, 106(1):37–43, 2008. doi:10.1016/j.ipl.2007.10.001.

15 Dion Harmon. *New Bounds on Optimal Binary Search Trees*. PhD thesis, Massachusetts Institute of Technology, 2006.

16 John Iacono. In pursuit of the dynamic optimality conjecture. In *Space-Efficient Data Structures, Streams, and Algorithms*, volume 8066 of *Lecture Notes in Computer Science*, pages 236–250. Springer Berlin Heidelberg, 2013. doi:10.1007/978-3-642-40273-9_16.

17 László Kozma. *Binary search trees, rectangles and patterns*. PhD thesis, Saarland University, Saarbrücken, Germany, 2016. URL: http://scidok.sulb.uni-saarland.de/volltexte/2016/6646/.

18 Victor Lecomte and Omri Weinstein. Settling the relationship between wilber's bounds for dynamic optimality. *arXiv preprint*, 2019. arXiv:1912.02858.

19 Joan M. Lucas. On the competitiveness of splay trees: Relations to the union-find problem. *On-line Algorithms, DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 7:95–124, 1991.

20 Seth Pettie. Splay trees, Davenport-Schinzel sequences, and the deque conjecture. In *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '08, pages 1115–1124, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1347082.1347204.

21 Daniel Dominic Sleator and Robert Endre Tarjan. Self-adjusting binary search trees. *J. ACM*, 32(3):652–686, 1985. Announced at STOC'83. doi:10.1145/3828.3835.

22 Rajamani Sundar. On the deque conjecture for the splay algorithm. *Combinatorica*, 12(1):95–124, 1992. doi:10.1007/BF01191208.

23 Robert Endre Tarjan. Sequential access in splay trees takes linear time. *Combinatorica*, 5(4):367–378, 1985. doi:10.1007/BF02579253.

24 Chengwen C. Wang, Jonathan C. Derryberry, and Daniel D. Sleator. O(log log n)-competitive dynamic binary search trees. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm*, SODA '06, pages 374–383, Philadelphia, PA, USA, 2006. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=1109557.1109600.

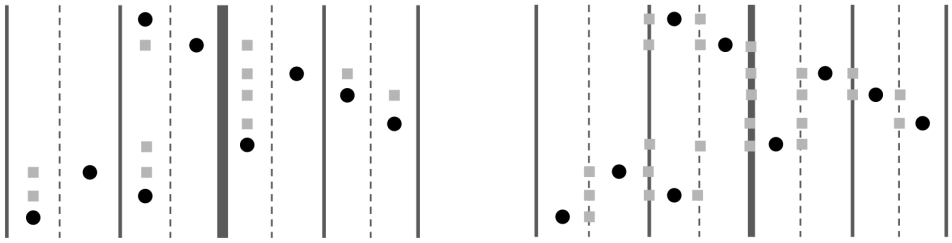
25 R. Wilber. Lower bounds for accessing binary search trees with rotations. *SIAM Journal on Computing*, 18(1):56–67, 1989. Announced at FOCS'86. doi:10.1137/0218004.

A A Detailed Description of the Algorithms

In this section we provide additional details for the proof of Theorem 2. Due to lack of space, some of the proofs are deferred to the full version.

A.1 Special Solutions

Our algorithm will produce feasible solutions of a special form, that we call *special solutions*. Recall that, given a semi-permutation point set X, the auxiliary columns for X are a set $\mathcal{L}$ of vertical lines with half-integral coordinates. We say that a solution Y for X is *special* iff every point of Y lies on an row that is active for X, and on a column of $\mathcal{L}$. In particular, special solutions are by definition non-canonical (see Figure 3 for an illustration). The main advantage of the special solutions is that they allow us to easily use the divide-and-conquer approach. We use the following observation, whose proof appears in the full version of the paper.



(a) Canonical Solution. **(b)** Special Solution.

Figure 3 Canonical and T-special solutions of X. The input points are shown as circles; the points that belong to the solution Y are shown as squares.

▶ **Observation 24.** *There is an algorithm, that, given a set X of points that is a semi-permutation, and a canonical solution Y for X, computes a special solution Y' for X, such that $|Y'| \leq 2|X| + 2|Y|$. The running time of the algorithm is $O(|X| + |Y|)$.*

If σ is any ordering of the auxiliary columns in $\mathcal{L}$, and $T = T_\sigma$ is the corresponding partitioning tree, then any point set Y that is a special solution for X is also called a *T-special solution* (although the notion of the solution Y being special does not depend on the tree T, this notion will be useful for us later; in particular, a convenient way of thinking of a T-special solution is that every point of Y must lie on an active row of X, and on a column that serves as a boundary for some strip $S(v)$, where $v \in V(T)$.)

A.2 Redundant Points and Reduced Point Sets

Consider a semi-permutation X, that we think of as a potential input to the Min-Sat problem. We denote $X = \{p_1, \ldots, p_m\}$, where the points are indexed in their natural bottom-to-top order, so $(p_1).y < (p_2).y < \ldots < (p_m).y$. A point p_i is said to be *redundant*, iff $(p_i).x = (p_{i+1}).x = (p_{i-1}).x$. We say that a semi-permutation X is in the *reduced form* if there are no redundant points in X; in other words, if p_{i-1}, p_i, p_{i+1} are three points lying on three consecutive active rows, then their x-coordinates are not all equal. We use the following observation and lemma, whose proofs appear in the full version of the paper.

▶ **Observation 25.** *Let X be a semi-permutation, and let $X' \subseteq X$ be any point set, that is obtained from X by repeatedly removing redundant points. Then $\mathsf{OPT}(X') \leq \mathsf{OPT}(X)$.*

▶ **Lemma 26.** *Let X be a semi-permutation, and let $X' \subseteq X$ be any point set, that is obtained from X by repeatedly removing redundant points. Let Y be any feasible solution for X' such that every point of Y lies on a row that is active for X'. Then Y is also a feasible solution for X.*

From Lemma 26, whenever we need to solve the Min-Sat problem on an instance X, it is sufficient to solve it on a sub-instance, obtained by iteratively removing redundant points from X. We obtain the following immediate corollary of Lemma 26.

▶ **Corollary 27.** *Let X be a semi-permutation, and let $X' \subseteq X$ be any point set, that is obtained from X by repeatedly removing redundant points. Let Y be any special feasible solution for X'. Then Y' is also a special feasible solution for X.*

Lastly, we need the following lemma, which is a simple application of the Wilber bound.

▶ **Lemma 28.** *Let X be a point set that is a semi-permutation in reduced form. Then $\mathsf{OPT}(X) \geq |X|/4 - 1$.*

A.3 The Algorithm Description

Suppose we are given an input set X of points that is a semi-permutation. Let T be any partitioning tree for X. We say that T is a *balanced* partitioning tree for X iff for every non-leaf vertex $v \in V(T)$ the following holds. Let v' and v'' be the children of v in the tree T. Let X' be the set of all input points lying in strip $S(v)$, and let X'', X''' be defined similarly for $S(v')$ and $S(v'')$. Let c be the number of active columns in instance X', and let c' and c'' be defined similarly for X'' and X'''. Then we require that $c', c'' \leq \lceil c/2 \rceil$.

Given a partitioning tree T, we denote by Λ_i the set of all vertices of T that lie in the ith *layer* of T – that is, the vertices whose distance from the root of T is i (so the root belongs to Λ_0). The *height* of the tree T, denoted by $\text{height}(T)$, is the largest index i such that $\Lambda_i \neq \emptyset$. If the height of the tree T is h, then we call the set $\Lambda_{\lceil h/2 \rceil}$ of vertices the *middle layer* of T. Notice that, if T is a balanced partitioning tree for input X, then its height is at most $2 \log c(X)$.

Our algorithm takes as input a set X of points that is a semi-permutation, a balanced partition tree T for X, and an integral parameter $\rho > 0$.

Intuitively, the algorithm uses the splitting operation to partition the instance X into subinstances that are then solved recursively, until it obtains a collection of instances whose corresponding partitioning trees have height at most ρ. We then either employ dynamic programming, or use a trivial $O(\log c(X))$-approximation algorithm. The algorithm returns a special feasible solution for the instance. Recall that the height of the tree T is bounded by $2 \log c(X) \leq 2 \log n$. The following two theorems will be used as the recursion basis.

▶ **Theorem 29.** *There is an algorithm called* LeafBST-1 *that, given a semi-permutation instance X of* Min-Sat *in reduced form, and a partitioning tree T for it, produces a feasible T-special solution for X of cost at most $2|X| + 2\mathsf{OPT}(X)$, in time $|X|^{O(1)} \cdot c(X)^{O(c(X))}$.*

▶ **Theorem 30.** *There is an algorithm called* LeafBST-2 *that, given a semi-permutation instance X of* Min-Sat *in reduced form, and a partitioning tree T for it, produces a feasible T-special solution of cost at most $2|X| \cdot \text{height}(T)$, in time $\text{poly}(|X|)$.*

The proofs of both theorems are deferred to the full version of the paper. We now provide a schematic description of our algorithm. Depending on the guarantees that we would like to achieve, whenever the algorithm calls procedure LeafBST, it will call either procedure LeafBST-1 or procedure LeafBST-2; we specify this later.

Algorithm 1 RecursiveBST(X, T, ρ).

1. Keep removing redundant points from X until X is in reduced form.
2. IF T has height at most ρ,
3. **return** LeafBST(X, T)
4. Let U be the set of vertices lying in the middle layer of T.
5. Compute the split $(X^c, \{X^s_v\}_{v \in U})$ of (X, T) at U.
6. Compute the corresponding sub-trees $(T^c, \{T^s_v\}_{v \in U})$ of T.
7. For each vertex $v \in U$, call to RecursiveBST with input (X^s_v, T^s_v, ρ), and let Y_v be the solution returned by it.
8. Call RecursiveBST with input (X^c, T^c, ρ), and let $\hat{Y}$ be the solution returned by it.
9. Let Z be a point set containing, for each vertex $v \in U$, for each point $p \in X^s_v$, two copies p' and p'' of p with $p'.y = p''.y = p.y$, where p' lies on the left boundary of $S(v)$, and p'' lies on the right boundary of $S(v)$.
10. **return** $Y^* = Z \cup \hat{Y} \cup (\bigcup_{v \in U} Y_v)$

A.4 Analysis

We start by showing that the solution that the algorithm returns is T-special in the following observation, whose proof appears in the full version of the paper.

▶ **Observation 31.** *Assuming that* LeafBST(X,T) *returns a* T*-special solution, the solution* Y^* *returned by Algorithm* RecursiveBST(X,T,ρ) *is a* T*-special solution.*

We next turn to prove that the solution Y^* computed by Algorithm RecursiveBST(X,T,ρ) is feasible. In order to do so, we will use the following immediate observation.

▶ **Observation 32.** *Let* Y^* *be the solution returned by Algorithm* RecursiveBST(X,T,ρ)*, and let* $u \in U$ *be any vertex. Then:*

- *Any point* $y \in Y^*$ *that lies in the interior of* $S(u)$ *must lie on an active row of instance* X^s_u*.*
- *Any point* $y \in Y^*$ *that lies on the boundary of* $S(u)$ *must belong to in* $\hat{Y} \cup Z$*. Moreover, the points of* $\hat{Y} \cup Z$ *may not lie in the interior of* $S(u)$*.*
- *If* R *is an active row for instance* X^s_u*, then set* Z *contains two points, lying on the intersection of* R *with the left and the right boundaries of* $S(u)$*, respectively.*

The following theorem, whose proof is deferred to the full version of the paper, shows that the algorithm returns a feasible solution.

▶ **Theorem 33.** *Assume that the recursive calls to Algorithm* RecursiveBST *return a feasible special solution* $\hat{Y}$ *for instance* X^c*, and for each* $v \in U$*, a feasible special solution* Y_v *for the strip instance* X^s_v*. Then the point set* $Y^* = Z \cup \hat{Y} \cup (\bigcup_{v \in U} Y_v)$ *is a feasible solution for instance* X*.*

In order to analyze the solution cost, consider the final solution Y^* to the input instance X. We distinguish between two types of pints in Y^*: a point $p \in Y^*$ is said to be of type 2 if it was added to the solution by Algorithm LeafBST, and otherwise we say that it is of type 1. We start by bounding the number of points of type 1 in Y^*.

▷ Claim 34. The number of points of type 1 in the solution Y^* to the original instance X is at most $O(\log(\text{height}(T)/\rho)) \cdot \mathsf{OPT}(X)$.

Proof. Observe that the number of recursive levels is bounded by $\lambda = O(\log(\text{height}(T)/\rho))$. This is since, in every recursive level, the heights of all trees decrease by a constant factor, and we terminate the algorithm once the tree heights are bounded by ρ. For each $1 \leq i \leq \lambda$, let $\mathcal{X}_i$ be the collection of all instances in the ith recursive level, where the instances are in the reduced form. Notice that the only points that are added to the solution by Algorithm RecursiveBST directly are the points in the sets Z. The number of such points added at recursive level i is bounded by $\sum_{X' \in \mathcal{X}_i} 2|X'|$. It is now sufficient to show that for all $1 \leq i \leq \lambda$, $\sum_{X' \in \mathcal{X}_i} |X'| \leq O(\mathsf{OPT}(X))$. We do so using the following observation.

▶ **Observation 35.** *For all* $1 \leq i \leq \lambda$, $\sum_{X' \in \mathcal{X}_i} \mathit{OPT}(X') \leq \mathit{OPT}(X)$.

Assume first that the observation is correct. For each instance $X' \in \mathcal{X}_i$, let T' be the partitioning tree associated with X'. From Lemma 28, $|X'| \leq O(\mathsf{OPT}(X'))$. Therefore, the number of type-1 points added to the solution at recursive level i is bounded by $O(\mathsf{OPT}(X))$. We now turn to prove Observation 35.

Proof of Observation 35. The proof is by induction on the recursive level i. It is easy to see that the claim holds for $i = 1$, since, from Observation 25, removing redundant points from X to turn it into reduced form cannot increase $\mathsf{OPT}(X)$.

Assume now that the claim holds for level i, and consider some level-i instance $X' \in \mathcal{X}_i$. Let $(X^c, \{X^s_u\}_{u \in U})$ be the split of (X', T') that we have computed. Then, from Theorem 12, $\sum_{v \in U} \mathsf{OPT}(X^s_v) + \mathsf{OPT}(X^c) \leq \mathsf{OPT}(X')$. Since, from Observation 25, removing redundant points from an instance does not increase its optimal solution cost, the observation follows. ◀

◁

In order to obtain an efficient $O(\log \log n)$-approximation algorithm, we set ρ to be a constant (it can even be set to 1), and we use algorithm LeafBST-2 whenever the algorithm calls to subroutine LeafBST. Observe that the depth of the recursion is now bounded by $O(\log \log n)$, and so the total number of type-1 points in the solution is bounded by $O(\log \log n) \cdot \mathsf{OPT}(X)$. Let $\mathcal{I}$ denote the set of all instances to which Algorithm LeafBST is applied. Using the same arguments as in Claim 34, $\sum_{X' \in \mathcal{I}} |X'| = O(\mathsf{OPT}(X))$. The number of type-2 points that Algorithm LeafBST adds to the solution for each instance $X' \in \mathcal{I}$ is bounded by $O(|X'| \cdot \rho) = O(|X'|)$. Therefore, the total number of type-2 points in the solution is bounded by $O(\mathsf{OPT}(X))$. Overall, we obtain a solution of cost at most $O(\log \log n) \cdot \mathsf{OPT}(X)$, and the running time of the algorithm is polynomial in $|X|$.

Finally, in order to obtain the sub-exponential time algorithm, we set the parameter ρ to be such that the recursion depth is bounded by D. Since the number of active columns in instance X is $c(X)$, and the height of the partitioning tree T is bounded by $2 \log c(X)$, while the depth of the recursion is at most $2 \log(\text{height}(T)/\rho)$, it is easy to verify that $\rho = O\left(\frac{\log c(X)}{2^{D/2}}\right) = \frac{\log c(X)}{2^{\Omega(D)}}$. We use algorithm LeafBST-1 whenever the algorithm calls to subroutine LeafBST. As before, let $\mathcal{I}$ be the set of all instances to which Algorithm LeafBST is applied. Using the same arguments as in Claim 34, $\sum_{X' \in \mathcal{I}} (|X'| + \mathsf{OPT}(X')) = O(\mathsf{OPT}(X))$. For each such instance X', Algorithm LeafBST-1 produces a solution of cost $O(|X'| + \mathsf{OPT}(X'))$. Therefore, the total number of type-2 points in the final solution is bounded by $O(\mathsf{OPT}(X))$. The total number of type-1 points in the solution is therefore bounded by $O(D) \cdot \mathsf{OPT}(X)$ as before. Therefore, the algorithm produces a factor-$O(D)$-approximate solution. Finally, in order to analyze the running time of the algorithm, we first bound the running time of all calls to procedure LeafBST-1. The number of such calls is bounded by $|X|$. Consider now some instance $X' \in \mathcal{I}$, and its corresponding partitioning tree T'. Since the height of T' is bounded by ρ, we get that $c(X') \leq 2^{\rho} \leq 2^{\log c(X)/2^{\Omega(D)}} \leq (c(X))^{1/2^{\Omega(D)}}$. Therefore, the running time of LeafBST-1 on instance X' is bounded by $|X'|^{O(1)} \cdot (c(X'))^{O(c(X'))} \leq |X'|^{O(1)} \cdot \exp\left(O(c(X') \log c(X')\right) \leq |X'|^{O(1)} \cdot \exp\left(c(X)^{1/2^{\Omega(D)}} \cdot \log c(X)\right)$.

The running time of the remainder of the algorithm, excluding the calls to LeafBST-1, is bounded by $\text{poly}(|X|)$. We conclude that the total running time of the algorithm is bounded by $|X|^{O(1)} \cdot \exp\left(c(X)^{1/2^{\Omega(D)}} \cdot \log c(X)\right) \leq \text{poly}(m) \cdot \exp\left(n^{1/2^{\Omega(D)}} \cdot \log n\right)$.

Revisiting Alphabet Reduction in Dinur's PCP

Venkatesan Guruswami
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
venkatg@cs.cmu.edu

Jakub Opršal
Computer Science Department, Durham University, UK
jakub.oprsal@durham.ac.uk

Sai Sandeep
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA
spallerl@andrew.cmu.edu

Abstract

Dinur's celebrated proof of the PCP theorem alternates two main steps in several iterations: gap amplification to increase the soundness gap by a large constant factor (at the expense of much larger alphabet size), and a composition step that brings back the alphabet size to an absolute constant (at the expense of a fixed constant factor loss in the soundness gap). We note that the gap amplification can produce a Label Cover CSP. This allows us to reduce the alphabet size via a direct long-code based reduction from Label Cover to a Boolean CSP. Our composition step thus bypasses the concept of Assignment Testers from Dinur's proof, and we believe it is more intuitive – *it is just a gadget reduction.* The analysis also uses only elementary facts (Parseval's identity) about Fourier Transforms over the hypercube.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness; Theory of computation → Approximation algorithms analysis

Keywords and phrases PCP theorem, CSP, discrete Fourier analysis, label cover, long code

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.34

Category APPROX

Funding *Venkatesan Guruswami*: Supported in part by NSF grants CCF-1526092, CCF-1814603, and CCF-1908125.
Jakub Opršal: Received funding from the UK EPSRC grant EP/R034516/1. Most of the work has been done during a visit to Carnegie Mellon University.
Sai Sandeep: Supported in part by NSF grant CCF-1908125.

Acknowledgements We would like to thank Irit Dinur, Margalit Glasgow, Oded Goldreich, Prahladh Harsha, Johan Håstad, Jaikumar Radhakrishnan, and Madhu Sudan for useful comments and feedback. We also thank the anonymous reviewers for valuable comments on the presentation.

1 Introduction

Constraint Satisfaction Problem (CSP) is a canonical NP-complete problem. Assuming P $\neq$ NP, no polynomial time algorithm can find a satisfying assignment to a satisfiable CSP instance. If we are happy with the easier goal of satisfying a $1 - o(1)$ fraction of constraints, does there exist an efficient algorithm to do so? Answering this in the negative, the fundamental PCP theorem [1, 2] implies that for some fixed integers $k, q \geq 2$ and $c < 1$, it is NP-hard to find an assignment satisfying a fraction c of constraints in a satisfiable CSP of arity k over alphabet $\{0, 1, \ldots, q-1\}$. Further this result holds for the combinations $(q, k) = (2, 3)$ and $(3, 2)$. The PCP theorem lies at the center of a rich body of work that has yielded numerous inapproximability results, including many optimal ones.


© Venkatesan Guruswami, Jakub Opršal, and Sai Sandeep;
licensed under Creative Commons License CC-BY



The PCP theorem was originally proved using algebraic techniques such as the low-degree test and the sum-check protocol. In a striking work, Dinur [7] gave an alternative combinatorial proof of the PCP theorem. Her proof works by amplifying the "Unsat value" of a CSP instance – the fraction of constraints any assignment should violate. The goal is to show that it is NP-hard to distinguish if the Unsat value of a CSP instance is equal to 0 or at least a constant $c > 0$. Starting with a NP-hard problem such as 3-coloring with m constraints, we can already deduce that it is NP-hard to identify if Unsat value is equal to 0 or at least $1/m$. The Unsat value is increased slowly and iteratively via two steps – *gap amplification* and *alphabet reduction*. In gap amplification, we incur a constant factor blow up in the size of the instance, and get a constant factor improvement in the Unsat value. However, this step also blows up the alphabet size. To alleviate this, alphabet reduction brings back the alphabet size to an absolute constant while losing a constant factor in the Unsat value (and blows up the instance size by a constant factor). Combining both the steps, we can increase the Unsat value by a constant factor (say 2) incurring a constant factor blow up in the size of the instance. Repeating this $\log m$ times proves the PCP theorem.

In this paper we revisit the alphabet reduction step. Dinur implemented this step by an "inner" PCP construction, which is in effect a gadget reducing a specific predicate ψ to be tested to a collection Ψ of constraints over a fixed (say Boolean) alphabet, such that if ψ is unsatisfiable, then a constant fraction of constraints of Ψ must be violated by any assignment.[1] This inner PCP is then applied to all constraints in the CSP instance (say $\mathcal{G}$) produced by the gap amplification step. The collection of inner PCPs as such only ensure that each constraint of $\mathcal{G}$ is individually satisfiable, which is not very meaningful. To ensure that the inner PCPs together ensure that the constraints of $\mathcal{G}$ are all satisfiable by a *single consistent* assignment, error-correcting codes are used to encode the purported assignments to the variables of $\mathcal{G}$. The inner PCP is also replaced by an *Assignment Tester* that ascertains whether the specific assignment given by these encodings satisfies the predicate ψ being checked.

The key observation driving this work is that instead of designing the inner PCP for *arbitrary* constraints (as in Dinur's paper), we can first reduce the CSP instance $\mathcal{G}$ produced by gap amplification to a *Label Cover* instance. Label Cover is a special kind of CSP which has arity 2, and whose underlying relations are functions (so the value of one of the variables in each constraint is determined by the value taken by the other variable in that constraint). Conveniently for us, we also observe that Dinur's gap amplification step in fact already produces a CSP with this Label Cover structure, allowing us to skip the reduction step.[2] We can thus focus on alphabet reduction when the CSP we are reducing from has the Label Cover structure, and is over a fixed, albeit large, alphabet. We then follow the influential Label Cover and Long Code framework, originally proposed in [5] and strengthened in [11] and since then applied in numerous works on inapproximability, to reduce the CSP obtained from gap amplification, now viewed as Label Cover, to a Boolean CSP. Finally, we reduce the Boolean CSP back to a Label Cover instance (see Section 4) that can be plugged in as input to the gap amplification step.

[1] While this might seem circular, as this is what the PCP reduction is trying to accomplish in the first place, the key is that this inner reduction can be highly inefficient (even triply exponential blow up is okay!), as it is applied to a constraint of constant size.

[2] Technically, the gap amplification step produces a version of Label Cover whose constraints are *rectangular* rather than functions, but this is a minor difference that can be easily accommodated in reductions from Label Cover.

Our main result is the following, which can be viewed as reproving a case of alphabet reduction from [8, 7].

▶ **Theorem 1.** *There is a polynomial time reduction from Label Cover with soundness* $1-\delta$ *to a fixed template CSP with soundness* $1-C\delta$ *for an absolute constant* $C>0$.

We analyze our reduction using Fourier analysis as pioneered by Håstad [11]. Usually, in this framework, we reduce from low soundness Label Cover to strong (and at times optimal) soundness of CSP. But here we start with a high soundness Label Cover, and we reduce to high soundness CSP.

We highlight a couple of differences from previous works that make our proof easier:

- We have the freedom to choose any CSP rather than trying to prove inapproximability of a CSP. We choose the following 4-ary predicate R in our reduction: $(u,v,w,x)\in R$ if and only if $u\neq v\vee w\neq x$. This predicate appears in [11] in the context of proving optimal hardness for NAE-4SAT.
- In [11] and [5], the objective is to prove optimal inapproximability, or at least to get good soundness. However, our present goal is to prove "just" a nontrivial soundness. (On the other hand, we also start with high soundness Label Cover.) This allows us to use a very convenient test distribution leading to a simple analysis.
- We remark that a similar statement as Theorem 1 can be also deduced using [5, Section 4.1.1]. We believe that the test presented in this paper is more direct since we benefit from ideas in [11].
- It is possible to perform alphabet reduction using the Hadamard code instead of the long code as described in [10, 12]; the latter [12], similarly to our proof, avoids explicit use of Assignment Testers.
- Long code tests correspond exactly to testing whether a function is a *polymorphism* of the corresponding CSP, and as such corresponds to gadget reductions in the algebraic approach to CSP (see e.g. [4]). The PCP theorem surpasses these algebraic (gadget) reductions; this is even more evident when extending the scope from CSPs to *promise constraint satisfaction problems* (PCSPs) as there are PCSPs that can be shown to be NP-hard by using PCP theorem via a natural reduction through Label Cover, but cannot be shown to be NP-hard using only algebraic reductions [3, 6]. In this sense, the present paper shows that this strength of the PCP theorem comes from the Gap Amplification step.

Alphabet Reduction is an essential step in both the original proof of the PCP theorem as well as Dinur's proof and deserves further attention. Our proof of alphabet reduction bypasses the concept of Assignment Testers and is more intuitive in our opinion as it is *nothing but a gadget reduction*. Our proof is elementary using only Parseval's identity from Fourier Analysis over the hypercube. Dinur's analysis used the Friedgut-Naor-Kalai theorem [9] about Boolean functions with most of the Fourier mass at level 1. We believe that this makes our proof more accessible to readers that are new to PCPs. We also hope that this material might be useful in teaching the proof of PCP theorem as it relies only on techniques that any such basic course would cover anyway.

Outline

We start by formally defining CSP, Label Cover, and other preliminaries in Section 2. Then, in Section 3, we prove the main reduction from Label Cover to CSP. In Section 4, we show how the reduction can be used in the alphabet reduction step of Dinur's proof.

2 Preliminaries

2.1 CSPs and Label Cover

Loosely speaking, a CSP over some domain Σ is a decision problem which gets as input a set of variables and a set of constraints on the values of these variables. The goal is to decide whether there is an assignment of values from Σ to the variables such that all the constraints are satisfied. Usually, the shape of the constraints is somehow restricted. We first give a formal definition of the general CSP, and then two restrictions that we will use further.

▶ **Definition 2** (CSP). *The constraint satisfaction problem over an alphabet Σ takes as input a set of variables $V = \{x_1, \ldots, x_n\}$ and a finite set of* constraints *where each constraint is a pair $((x_{i_1}, \ldots, x_{i_k}), R)$ where k is a number (the arity of the constraint), $i_1, \ldots, i_k \in \{1, \ldots, n\}$, and $R \subseteq \Sigma^k$. The goal is to decide whether there exists an assignment $s\colon V \to \Sigma$ such that for each constraint $((x_{i_1}, \ldots, x_{i_k}), R)$ we have $(s(x_{i_1}), \ldots, s(x_{i_k})) \in R$.*

A fixed template CSP is a restriction of the general CSP that requires that the constraints only involve relations from a fixed list of relations over the given alphabet (a template). In the case the domain is Boolean, often negation of variables is allowed. Below, we formally define a Boolean fixed template CSP with a template consisting of a single relation allowing for negation of variables.

▶ **Definition 3** (Boolean fixed template CSP). *Let $\Sigma = \{0, 1\}$ be a Boolean domain, and fix a relation $R \subseteq \Sigma^k$. The constraint satisfaction problem associated with R, denoted by $\mathrm{CSP}(R)$, takes input as a set of variables V and a set of constraints of the form $((x_{i_1}, \ldots, x_{i_k}), R)$ where each x_i is either a variable x, or a negation of a variable (i.e., an expression $\neg x$). An assignment $s\colon V \to \Sigma$ is said to satisfy the constraint $((x_{i_1}, \ldots, x_{i_k}), R)$ if $(s(x_{i_1}), \ldots, s(x_{i_k})) \in R$ where $s(\neg x)$ is defined as $\neg s(x)$ for each $x \in V$.*

The restriction of CSP to binary constraints is traditionally referred to as Label Cover.

▶ **Definition 4** (Label Cover). *In an instance of Label Cover problem, we are given a tuple $(G = (V, E), \Sigma, \Pi)$ where*

1. *G is a graph on vertex set V.*
2. *Each vertex in G has to be assigned a label from Σ.*
3. *For each edge $e = (u, v) \in E$, there is a relation $\Pi_e \subseteq \Sigma \times \Sigma$. This relation corresponds to a constraint between u and v.*

A labeling of graph is a function $s\colon V \to \Sigma$ that assigns a label to each vertex of G. Such labeling is said to satisfy a constraint e if and only if $(s(u), s(v)) \in \Pi_e$.

For a Label Cover instance or in general for a CSP instance I, we use $\mathrm{size}(I)$ to denote $m + n$, where m is the number of constraints and n is the number of variables. We remark that Label Cover usually refers to the case when G is bipartite, and the constraint relations are functions. However, in this work, we find it convenient to consider a (closely related) version which has *rectangular relations*.

▶ **Definition 5** (Rectangular relation). *A relation $R \subseteq A \times B$ is said to be rectangular if there is a set C and functions $\pi\colon A \to C$ and $\sigma\colon B \to C$ such that $(a, b) \in R$ if and only if $\pi(a) = \sigma(b)$. Equivalently, R is rectangular if for all $a, a' \in A$ and $b, b' \in B$ such that $(a, b) \in R$, $(a, b') \in R$, and $(a', b) \in R$, we have $(a', b') \in R$.*

2.2 The long code

Loosely speaking, the *long code* is the longest (error-correcting) code over the Boolean alphabet that does not repeat bits. It is constructed as follows: the long code is a Boolean code of length 2^n which encodes a value $i \in [n]$ into a tuple p_i whose k-th coordinate, $k < 2^n$, is the i-th least significant digit of k in binary.

The long code be also described in another way: we view a Boolean tuple of length 2^n as an n-ary function $p\colon \{0,1\}^n \to \{0,1\}$ (each coordinate of the tuple encodes one value of p). In this perspective, the code words of the long code are functions p_i defined as $p_i(x_1,\dots,x_n) = x_i$. These functions are often called *dictators*.

We also remark that in the conjunction with the long-code, a rectangular constraint can be expressed as an identity. More precisely, given a rectangular relation $R \subseteq [n] \times [m]$, say $R = \{(i,j) : \pi(i) = \sigma(j)\}$ for some $\pi\colon [n] \to [k]$ and $\sigma\colon [m] \to [k]$, then the long codes p_i and p_j of values i, j satisfy

$$p_i(x_{\pi(1)},\dots,x_{\pi(n)}) = p_j(x_{\sigma(1)},\dots,x_{\sigma(m)})$$

for all $x_1,\dots,x_k \in \{0,1\}$ if and only if $(i,j) \in R$. This is a key property of rectangular relations that is used implicitly in our proof.

2.3 Boolean Fourier analysis

As usual in Boolean Fourier analysis, we treat TRUE as -1 and FALSE as $+1$. In particular, in this notation, "negation" is expressed as $\neg x = -x$, "xor" $x \oplus y$ is expressed as $x \oplus y = xy$, and "or" is the expressed by the following function:

$$x \vee y = \begin{cases} -1 & \text{if } x = -1 \text{ or } y = -1, \text{ and} \\ 1 & \text{otherwise.} \end{cases}$$

Throughout the paper, we will use all the same symbols to denote the coordinatewise (or bitwise) application of these functions to tuples, e.g. $(x_1,x_2) \oplus (y_1,y_2) = (x_1y_1, x_2y_2)$.

We define an inner product space on functions from $\{-1,+1\}^n \to \mathbb{R}$ as $\langle f,g\rangle = \mathbb{E}_x[f(x)g(x)]$. For a set $\alpha \subseteq [n]$, let

$$\chi_\alpha(x_1,\dots,x_n) = \prod_{i\in\alpha} x_i.$$

The set of such functions form an orthonormal basis for all functions from $\{-1,+1\}^n$ to $\mathbb{R}$ in the above defined inner product space. Moreover, if $\alpha \neq \emptyset$, then $\mathbb{E}_x[\chi_\alpha(x)] = 0$.

▶ **Definition 6** (Fourier expansion). *Given a function $f\colon \{-1,+1\}^n \to \mathbb{R}$, we can thus write it uniquely as a linear combination of this basis –*

$$f = \sum_{\alpha\subseteq[n]} \hat{f}(\alpha)\chi_\alpha$$

The real quantities $\hat{f}(\alpha)$ are called the Fourier coefficients *of f. We abuse the notation $\hat{f}(i)$ to denote $\hat{f}(\{i\})$.*

The following simple but crucial identity follows from the definitions and is all that we will need in our analysis.

▶ **Theorem 7** (Parseval's Identity). *For each Boolean valued function f, i.e., $f\colon \{-1,+1\}^n \to \{-1,+1\}$,*

$$\sum_{\alpha\subseteq[n]} \hat{f}(\alpha)^2 = 1.$$

2.3.1 Connection to the long code

We remark, that the function $\chi_{\{i\}}$ corresponds to a valid long code: the function p_i encoding the value i. Also observe that there is a connection between the natural distance defined by the inner product $\langle f, g\rangle$ on Boolean functions and *relative Hamming distance* of f and g: This is thanks to the fact that if $x, y \in \{-1, 1\}$ then $x = y$ if and only if $xy = 1$, and consequently, the relative Hamming distance of f to the long code word $p_i = \chi_{\{i\}}$ can be expressed as $\big(1 - \hat{f}(i)\big)/2$. This means that the closest valid long code to a function f is the p_i for which $\hat{f}(i)$ is maximal.

These ideas are manifested in the common strategy in rounding a Boolean function f to a long code: First make sure that coefficients $\hat{f}(\alpha)$ for large sets α are small enough, then decode to a value i that belongs to a small-enough (ideally 1-element) set α with a large-enough $\hat{f}(\alpha)$.

3 Label Cover to CSP

This section describes our gadget reduction from Label Cover to CSP(R) where R is the 4-ary relation over Boolean domain defined as

$$R = \{(x, x', z, z') \mid x \neq x' \vee z \neq z'\}.$$

▶ **Theorem 8.** *There exists absolute constant C such that given a Label Cover instance (not necessarily bipartite) $G = (V, E, \Sigma, \Pi)$ with rectangular constraints, there is a reduction from G that outputs an instance I of* CSP(R) *such that* size(I) $= O($size($G)$) *and*

- *If G is satisfiable, then I is satisfiable as well.*
- *If no labeling can satisfy $1 - \delta$ fraction of constraints of G, then no assignment can satisfy $1 - C\delta$ fraction of constraints in I for all δ.*

Since the domain of CSP(R) is Boolean, the above reduces from an alphabet Σ of arbitrary size to the alphabet of size 2. We note that the constant in $O($size$(G))$ above hides exponential dependency on $|\Sigma|$.

We describe the reduction as a probabilistic checker of a solution to G encoded using a long code, i.e., the proof contains for each $u \in V$ a word $f_u \colon \{-1, 1\}^{|\Sigma|} \to \{-1, 1\}$. In other words, we design the test in such a way that if $s \colon V \to \Sigma$ is a solution to G, then the assignment $f_u \mapsto p_{s(u)}$ passes the test. This will then immediately give the completeness of the reduction. The test is as follows: Sample an edge $e = (u, v)$ from E uniformly at random, and then with equal probability do one of the following

1. run a long code test inside f_u;
2. run a long code test inside f_v;
3. run a constraint test between f_u and f_v.

We describe the long code test and the constraint test below. Both query the respective tables of f_u and f_v at some 4 bits that are generated by a certain randomized algorithm, and then check whether these 4 Boolean values satisfy the predicate R defined above.

This checker can be viewed as a gadget reduction in the following sense: We replace each vertex $u \in V$ with $2^{|\Sigma|}$ Boolean variables labeled by $f_u(x)$ for $x \in \{-1, 1\}^{|\Sigma|}$ (we see an assignment to such variables as a function $f_u \colon \{-1, 1\}^{|\Sigma|} \to \{-1, 1\}$), and each edge $e = (u, v)$ with a set of weighted 4-ary constraints on f_u and f_v, each involving the relation R and some 4 values of f_u and f_v (the result is therefore an instance of CSP(R)). These constraints depend only on the relation Π_e.

To simplify some notation, we assume $\Sigma = [n]$. We also assume that the tables for f_u's are *folded* so f_u is forced to satisfy $f(-x) = -f(x)$. This is a standard technique. Such a folding is ensured by including only one variable of each pair $x, -x$, and if the test queries f_u at the bit corresponding to some x that is not included, the bit $f(-x)$ is queried instead, and the value is negated. As a consequence of this folding, we have to allow for negation of variables in CSP(R). An important and useful consequence of this is that all "even" Fourier coefficients of f vanish, i.e., $\hat{f}(\beta) = 0$ for all β such that $|\beta|$ is even. We remark that folding can be avoided in the construction of the gadget. Nevertheless, it considerably simplifies the calculations below. Further, for calculations, it is useful to view R as a predicate $\rho\colon \{\pm 1\}^4 \to \{0,1\}$ defined as

$$\rho(x,x',z,z') = 1 - \frac{(xx'+1)(zz'+1)}{4}.$$

It is easy to check that $\rho(x,x',z,z') = 1$ if and only if $(x,x',z,z') \in R$.

Let us now describe the two probabilistic checkers.

3.1 Long code test

The long code test has on input a table of a function f ($= f_u$ or f_v), and it is supposed to check whether this function is a code word of the long code, i.e., there is i such that $f = p_i$. We design the test so that all these words pass with probability 1. Since we are only using the predicate R, this further limits possible checks. In fact, we include all checks of the form $f(x_1,x_2,x_3,x_4) \in R$ that are passed by all dictators.[3]

Long code test. Given $f\colon \{-1,1\}^n \to \{-1,1\}$ to test against being a long code word. Choose $x,y,z,\mu \in \{-1,1\}^n$ uniformly at random. Test whether

$$(f(x), f(x \oplus (\mu \vee y)), f(z), f(z \oplus (\mu \vee \neg y))) \in R. \tag{1}$$

Note that for all $x,y,z,\mu \in \{-1,1\}$, $(x, x \oplus (\mu \vee y), z, z \oplus (\mu \vee \neg y)) \in R$. This implies that any dictator function passes the test with probability 1, and therefore provides the completeness of the test. We also note that this test can give some false positives, e.g. the function $-p_i\colon x \mapsto -p_i(x)$ passes the test with probability 1, but is not a long code word. It is in fact a negation of the word p_i. It can be checked that all functions that pass are either long code words, or their negations. In the decoding, we simply decode the above function f to i.

The following lemma bounds the probability that the test accepts in the means of the Fourier coefficients. We remark, that since we want to ensure that f is as close to a valid long code as possible, the probability should decrease as the coefficients $\hat{f}(\alpha)$ for $\alpha \neq \{i\}$ increase. Indeed, the lemma states that this is the case.

▶ **Lemma 9.** *Assuming that f is folded, the probability the long code test accepts is at most*

$$1 - \frac{3}{16} \sum_{|\alpha|>1} \hat{f}(\alpha)^2.$$

[3] Any function that passes any such test with probability 1 is called a *polymorphism of R*, see also [4].

Proof. Assume $f(x) = \sum_\alpha \hat{f}(\alpha)\chi_\alpha(x)$. The probability that the test accepts is

$$\begin{aligned}
&\mathbb{E}_{x,y,z,\mu}\, \rho(f(x),f(x \oplus (\mu \vee y)), f(z), f(z \oplus (\mu \vee \neg y))) \qquad (2)\\
&\quad = \mathbb{E}_{x,y,z,\mu}\left[1 - \frac{\big(f(x)f(x \oplus (\mu \vee y) + 1\big)\big(f(z)f(z \oplus (\mu \vee \neg y)) + 1\big)}{4}\right]\\
&\quad = \frac{3}{4} - \frac{1}{4}\,\mathbb{E}_{x,y,\mu}\, f(x)f(x \oplus (\mu \vee y)) - \frac{1}{4}\,\mathbb{E}_{y,z,\mu}\, f(z)f(z \oplus (\mu \vee \neg y))\\
&\qquad - \frac{1}{4}\,\mathbb{E}_{x,y,z,\mu}\, f(x)f(x \oplus (\mu \vee y))f(z)f(z \oplus (\mu \vee \neg y))
\end{aligned}$$

We further simplify this expression one term at a time.

$$\begin{aligned}
\mathbb{E}_{x,y,\mu}\, f(x)f(x \oplus (\mu \vee y)) &= \mathbb{E}_{x,y,\mu} \sum_{\alpha,\beta} \hat{f}(\alpha)\hat{f}(\beta)\chi_\alpha(x)\chi_\beta(x \oplus (\mu \vee y)) \qquad (3)\\
&= \sum_{\alpha,\beta} \hat{f}(\alpha)\hat{f}(\beta)\,\mathbb{E}_x[\chi_\alpha(x)\chi_\beta(x)]\,\mathbb{E}_{y,\mu}[\chi_\beta(\mu \vee y)]\\
&= \sum_{\alpha} \hat{f}(\alpha)^2\,\mathbb{E}_{y,\mu}[\chi_\alpha(y \vee \mu)] = \sum_{\alpha} \hat{f}(\alpha)^2(-1/2)^{|\alpha|}.
\end{aligned}$$

The third equality follows since χ_α and χ_β are orthogonal if $\alpha \neq \beta$. The last equality follows from the fact that $\mathbb{E}_{y,\mu}[y \vee \mu] = (-1) \cdot 3/4 + 1 \cdot 1/4 = -1/2$ and coordinates are chosen independently. Similarly, we get that

$$\mathbb{E}_{y,z,\mu}\, f(z)f(z \oplus (\mu \vee \neg y)) = \sum_{\alpha} \hat{f}(\alpha)^2(-1/2)^{|\alpha|}. \qquad (4)$$

Moving to the next term, we get

$$\begin{aligned}
&\mathbb{E}_{x,y,z,\mu}\, f(x)f(x \oplus (\mu \vee y))f(z)f(z \oplus (\mu \vee \neg y))\\
&\quad = \sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{f}(\beta)^2\,\mathbb{E}_{y,\mu}\,\chi_\alpha(\mu \vee y)\chi_\beta(\mu \vee \neg y) = \sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(-1/2)^{|\alpha \cup \beta|}. \qquad (5)
\end{aligned}$$

The last equality follows since $\mathbb{E}_{y,\mu}[(\mu \vee y) \oplus (\mu \vee \neg y)] = \mathbb{E}_{y,\mu}[\neg\mu] = 0$ and $\mathbb{E}_{y,\mu}[\mu \vee y] = \mathbb{E}_{y,\mu}[\mu \vee \neg y] = -1/2$. The overall acceptance probability is then

$$\begin{aligned}
&\frac{3}{4} - \frac{1}{2}\sum_{\alpha} \hat{f}(\alpha)^2(-1/2)^{|\alpha|} - \frac{1}{4}\sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(-1/2)^{|\alpha \cup \beta|}\\
&\qquad = 1 - \frac{1}{2}\sum_{\alpha} \hat{f}(\alpha)^2\big((-1/2)^{|\alpha|} + 1/2\big) - \frac{1}{4}\sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(-1/2)^{|\alpha \cup \beta|} \qquad (6)
\end{aligned}$$

where for the last equality, we used Parseval's identity. Further, we assumed that f is folded, and therefore $\hat{f}(\alpha) = 0$ for all α such that $|\alpha|$ is even. Restricting the sums to α and β with odd cardinality, and using that for such disjoint α and β, $|\alpha \cup \beta|$ is even, the last expression of (6) can be bounded from above by

$$1 - \frac{1}{2}\sum_{|\alpha|>1} \hat{f}(\alpha)^2\,(3/8) - \frac{1}{4}\sum_{\alpha \cap \beta = \emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(1/2)^{|\alpha \cup \beta|} \leq 1 - \frac{3}{16}\sum_{|\alpha|>1} \hat{f}(\alpha)^2 \qquad (7)$$

which concludes the proof. ◀

3.2 Constraint test

The constraint test has on input tables for functions f and g corresponding to some u and v such that $(u, v) \in E$, and it is supposed to test (assuming f and g are correct long code words) whether the values these functions encode satisfy the constraint given by a rectangular relation Π_e. We construct the test in a similar way as the long code test: We test functions f and g in 4 bits in such a way that long code words encoding satisfying values pass. In contrast with the long code test, we do not include all such tests, but only a selection; in particular, we include only tests that query two values from each function.

We assume that the constraint relation Π_e is given by $\pi, \sigma \colon [n] \to [m]$ such that $(i, j) \in \Pi_e$ if and only if $\pi(i) = \sigma(j)$, and we denote by y^π the vector in $\{-1, 1\}^n$ such that $y^\pi(i) = y(\pi(i))$.

Constraint test. Given $f, g \colon \{-1, 1\}^n \to \{-1, 1\}$ to test against satisfying a constraint Π_e given by $(i, j) \in \Pi_e$ if and only if $\pi(i) = \sigma(j)$ for fixed $\pi, \sigma \colon [n] \to [m]$. Choose $x, z \in \{-1, 1\}^n$ and $y \in \{-1, 1\}^m$ uniformly at random, and test whether

$$(f(x), f(x \oplus y^\pi), g(z), g(z \oplus (\neg y)^\sigma)) \in R. \tag{8}$$

Note that if both f and g are dictators, say $f = p_i$ and $g = p_j$, such that $\pi(i) = \sigma(j) = k$ then the above test accepts with probability 1. Indeed, the tuple gets evaluated to

$$(x(i), (x \oplus y^\pi)(i), z(j), (z \oplus (\neg y)^\sigma)(j)) = (x(i), x(i) \oplus y(k), z(j), z(j) \oplus \neg y(k))$$

which is in R for all x, y and z. This provides the completeness of the test.

In the analysis below, we will use the following notation.

▶ **Definition 10.** *Let $\alpha \subseteq [n]$ and $\pi \colon [n] \to [m]$, we denote by $\pi[\alpha]$ the subset of $[m]$ defined by $\pi[\alpha] = \{k : |\pi^{-1}(k) \cap \alpha| \text{ is odd}\}$.*

The goal of the constraint check is to ensure that functions f, g which are far from valid long codes that encode values satisfying the constraint pass with low probability. Unfortunately, the test gives a lot of false positives: it accepts any pair of functions χ_α and χ_β such that $\pi[\alpha] = \sigma[\beta]$ with probability 1.[4] This is nevertheless good-enough since the long code test provides that relevant α and β contain only one element, and $\pi[\{i\}] = \sigma[\{j\}]$ if and only if $\pi(i) = \sigma(j)$.

Naturally, the pairs of α and β with $\pi[\alpha] = \sigma[\beta]$ will appear in the analysis below. A useful fact that will simplify the computation below is that $\prod_{i \in \alpha} x_{\pi(i)} = \prod_{i \in \pi[\alpha]} x_i$, for all $x_1, \ldots, x_m \in \{-1, 1\}$, which implies that

$$\chi_\alpha(x^\pi) = \chi_{\pi[\alpha]}(x).$$

▶ **Lemma 11.** *Given that both f and g are folded, the probability that the consistency test accepts is at most*

$$1 - \frac{1}{4} \sum_{i,j : \pi(i) \neq \sigma(j)} \hat{f}(i)^2 \hat{g}(j)^2.$$

[4] We note that $\pi[\alpha] = \sigma[\beta]$ is equivalent to $\chi_\alpha(x_{\pi(1)}, \ldots, x_{\pi(n)}) = \chi_\beta(x_{\sigma(1)}, \ldots, x_{\sigma(n)})$ for all $x \in \{-1, 1\}^m$.

Proof. We can compute the acceptance probability in the same way as before, i.e., as

$$\frac{3}{4} - \frac{1}{4}\,\mathbb{E}_{x,y}[f(x)f(x \oplus y^\pi)] - \frac{1}{4}\,\mathbb{E}_{z,y}[g(z)g(z \oplus (\neg y)^\sigma)] \\ - \frac{1}{4}\,\mathbb{E}_{x,y,z}[f(x)f(x \oplus y^\pi)g(z)g(z \oplus (\neg y)^\sigma)] \quad (9)$$

We have

$$\mathbb{E}_{x,y}[f(x)f(x \oplus y^\pi)] = \sum_\alpha \hat{f}(\alpha)^2\,\mathbb{E}_y[\chi_\alpha(y^\pi)] = \sum_\alpha \hat{f}(\alpha)^2\,\mathbb{E}_y[\chi_{\pi[\alpha]}(y)] = 0 \quad (10)$$

where the last equality holds since $|\alpha|$ is odd, and consequently $\pi[\alpha] \neq \emptyset$. Similarly, $\mathbb{E}_{x,z}[g(z)g(z \oplus (\neg y)^\sigma)]$ vanishes. Thus the probability that the test accepts is

$$\begin{aligned} &\frac{3}{4} - \frac{1}{4}\,\mathbb{E}_{x,y,z}\, f(x)f(x \oplus y^\pi)g(z)g(z \oplus (\neg y)^\sigma) \quad (11)\\ &\qquad = \frac{3}{4} - \frac{1}{4}\sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{g}(\beta)^2\,\mathbb{E}_y[\chi_\alpha(y^\pi)\chi_\beta(-y^\sigma)] \\ &\qquad = \frac{3}{4} + \frac{1}{4}\sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{g}(\beta)^2\,\mathbb{E}_y[\chi_\alpha(y^\pi)\chi_\beta(y^\sigma)] \\ &\qquad = \frac{3}{4} + \frac{1}{4}\sum_{\alpha,\beta} \hat{f}(\alpha)^2\hat{g}(\beta)^2\,\mathbb{E}_y[\chi_{\pi[\alpha]}(y)\chi_{\sigma[\beta]}(y)] \\ &\qquad = \frac{3}{4} + \frac{1}{4}\sum_{\alpha,\beta:\pi[\alpha]=\sigma[\beta]} \hat{f}(\alpha)^2\hat{g}(\beta)^2 \\ &\qquad = 1 - \frac{1}{4}\sum_{\alpha,\beta:\pi[\alpha]\neq\sigma[\beta]} \hat{f}(\alpha)^2\hat{g}(\beta)^2 \end{aligned}$$

where the second equality follows from $|\beta|$ being odd, and the last equality follows from the Parseval's identity. Since $\pi(i) = \sigma(j)$ implies that $\pi[\{i\}] = \sigma[\{j\}]$, the claim follows. ◀

3.3 The full test

Putting the analysis of the two tests together we get the following.

▶ **Lemma 12.** *Given that both f and g are folded, the probability that the joint test accepts is at most*

$$1 - \frac{1}{16}\Big(\sum_{|\alpha|>1} f(\alpha)^2 + \sum_{|\beta|>1} g(\beta)^2 + \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2\Big)$$

Proof. Follows directly from Lemmas 9 and 11. ◀

Finally, we are ready to prove the main theorem of this section.

Proof of Theorem 8. The completeness follows in a straightforward way from the two comments after the description of the tests. We prove the soundness. Suppose that the test passes with probability $1 - \delta$. We will show that this implies that there is an assignment to the Label Cover instance that satisfies $(1 - 16\delta)$-fraction of constraints.

Our decoding is as follows: for a node $v \in V$, decode v to $i \in \Sigma$ with probability proportional to $\hat{f}_v(i)^2$. Intuitively, we decode to the value i with higher probability if f is closer to the code word $p_i = \chi_{\{i\}}$ or its negative $-p_i$ (see also Section 2.3.1). We will show that in expectation, this decoding satisfies at least $1 - 16\delta$ fraction of constraints, which proves that there exists a labeling that satisfies at least $1 - 16\delta$ fraction of constraints.

Let $1-\delta_e$ denote the probability that the test passes when we pick edge e. As test passes with probability $1-\delta$, we know that $\mathbb{E}_e[\delta_e] = \delta$. Suppose that we pick $e = (u, v)$ with f and g being the functions corresponding to u and v respectively. From the above lemma, we have that

$$1 - \delta_e \leq 1 - \frac{1}{16}\Big(\sum_{|\alpha|>1} \hat{f}(\alpha)^2 + \sum_{|\beta|>1} \hat{g}(\beta)^2 + \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2\Big), \tag{12}$$

and therefore,

$$16\delta_e \geq \sum_{|\alpha|>1} \hat{f}(\alpha)^2 + \sum_{|\beta|>1} \hat{g}(\beta)^2 + \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2. \tag{13}$$

The probability that our decoding satisfies edge e of Label Cover is at least

$$\begin{aligned}
\sum_{i,j:\pi(i)=\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2 &= 1 - \sum_{\substack{\alpha,\beta \\ |\alpha|>1, \text{ or } |\beta|>1, \text{ or } \alpha=\{i\} \text{ and } \beta=\{j\} \text{ and } \pi(i)\neq\sigma(j)}} \hat{f}(\alpha)^2\hat{g}(\beta)^2 \qquad (14)\\
&\geq 1 - \sum_{\substack{\alpha,\beta \\ |\alpha|>1}} \hat{f}(\alpha)^2\hat{g}(\beta)^2 - \sum_{\substack{\alpha,\beta \\ |\beta|>1}} \hat{f}(\alpha)^2\hat{g}(\beta)^2 - \sum_{\substack{i,j \\ \pi(i)\neq\sigma(j)}} \hat{f}(i)^2\hat{g}(j)^2\\
&= 1 - \sum_{|\alpha|>1} \hat{f}(\alpha)^2 - \sum_{|\alpha|>1} \hat{g}(\alpha)^2 - \sum_{i,j:\pi(i)\neq\sigma(j)} \hat{f}(i)^2\hat{g}(j)^2\\
&\geq 1 - 16\delta_\epsilon
\end{aligned}$$

where the first equality follows from Perseval's identity. Thus, the expected number of constraints satisfied by the labeling is at least $\mathbb{E}_e[1 - 16\delta_e] = 1 - 16\delta$ which proves the required claim with $C = 1/16$. ◀

Theorem 1 is stated without the assumption that the constraints are rectangular. This slightly more general version follows from Theorem 8 by a standard reduction which we describe below, in the proof of Theorem 13.

4 CSP to Label Cover

In this section, we recall the basic structure of Dinur's proof of PCP Theorem, and show how the previous reduction can be used in the alphabet reduction step of Dinur's proof. The resulting proof requires a gap amplification step for which we refer to Dinur's paper [7].

We first prove that the previous reduction can be combined with standard reductions to get back Label Cover from the CSP.

▶ **Theorem 13** (Alphabet reduction). *Given a Label Cover instance $G = (V, E, \Pi, \Sigma_0)$ with rectangular constraints, there is a polynomial time reduction that outputs another Label Cover instance with rectangular constraints $G' = (V', E', \Pi', \Sigma)$ with alphabet size Σ such that $|\Sigma|$ is an absolute constant, $size(G') = O(size(G))$ and*

- *If G is satisfiable, then G' is satisfiable as well.*
- *If every labeling violates δ fraction of constraints of G, then every labeling violates $C\delta$ fraction of constraints in G' for an absolute constant C.*

Proof. We first convert the Label Cover instance G to a CSP instance I as in Theorem 8. The CSP instance can be converted to bipartite Label Cover using standard clause-variable Label-coverization technique. We include the proof here for the sake of completeness. We

have n vertices $x_1, x_2, \ldots, x_n$ corresponding to the variables of I on the left L, and m vertices corresponding to constraints $C_1, C_2, \ldots, C_m$ of I on the right R. The label set is binary on the left, and satisfying assignments (at most 16) on the right corresponding to the possible assignments to four variables in the constraint. We add an edge between $u \in L$ and $v \in R$ if $x_u \in C_v$. The constraint on this edge enforces that the assignment to x_u is consistent with the assignment C_v assigns to x_u.

If there is a satisfying labeling to G, there is a satisfying assignment to I. Using this, we can assign the variables on the left the satisfying assignment, and the corresponding assignment to tuples for the vertices of constraints on the right, and thus get a satisfying assignment to G'. Suppose that every labeling violates at least δ fraction of constraints of G. From Theorem 8, every assignment violates at least $C\delta$ fraction of constraints in I. Suppose there is a labeling to G' that satisfies δ' fraction of constraints. Consider the assignment obtained by this labeling on the left. This assignment violates at least $C\delta m$ number of constraints in I. Note that this should violate at least $C\delta m$ constraints in G' and thus $\delta' \geq C'\delta$ for an absolute constant C'. The constraints are in fact projections, and thus are rectangular too. ◀

In order for us to use this as Composition step in the PCP of Dinur, we need the final observation that the output of Gap Amplification applied to a CSP with rectangular constraints results in a Label Cover with rectangular constraints. Dinur achieves gap amplification by "graph powering" which is described more formally below.

▶ **Definition 14** (Constraint Graph Powering). *Given a d-regular Label Cover (a.k.a. Constraint graph) $G = (V, E, \Sigma, \Pi)$, we obtain t-th power of it $G^t = (V, E', \Sigma', \Pi')$ as follows:*

- *Vertices. The vertices in G^t are the same as vertices in G.*
- *Edges. u and v are connected by k parallel edges in E' if there are exactly k paths of length t between u and v in G.*
- *Alphabet. The alphabet of G^t is $\Sigma^{d^{\lceil t/2 \rceil}}$. A value $a \in \Sigma^{d^{\lceil t/2 \rceil}}$ is interpreted as assigning values $a \colon \Gamma(u) \to \Sigma$ to $d^{\lceil t/2 \rceil}$ elements from Σ. This value is treated as u's opinion on $\Gamma(u)$, the set of all vertices within $\lceil t/2 \rceil$ distance from u.*
- *Constraints. An edge $(u, v) \in E'$ is satisfied by $a, b \in \Sigma^{d^{\lceil t/2 \rceil}}$ if and only if the following holds: there is an assignment $\sigma \colon \Gamma(u) \cup \Gamma(v) \to \Sigma$ that satisfies every constraint $c(e)$ where $e \in E \cap (\Gamma(u) \times \Gamma(v))$, and such that*

$$\forall u' \in \Gamma(u), \sigma(u) = a_{u'}; \forall v' \in \Gamma(v), \sigma(v) = b_{v'}$$

where $a_{u'}$ (and respectively $b_{v'}$) is the value a (and resp. b) assigned to u' (and resp. v').

The output G^t is also a binary CSP, and hence can be viewed as a Label Cover. We claim that if every constraint of G is rectangular, then every constraint of G^t is rectangular as well. Let $e = (u, v)$ be an edge in E' with constraint relation as R_e. Suppose $(a, b), (a', b), (a, b') \in R_e$. This implies that for all $(u', v') \in E \cap (\Gamma(u) \times \Gamma(v))$ with constraint relation c_e,

$$(a_{u'}, b_{v'}), (a'_{u'}, b_{v'}), (a_{u'}, b'_{v'}) \in R_{c_e}.$$

Since R_{c_e} is rectangular, $(a'_{u'}, b'_{v'}) \in R_{c_e}$ as well. As this holds for all such u' and v', $(a', b') \in R_e$, thus proving that R_e is a rectangular relation.

Combined with the preprocessing step, the gap amplification theorem of Dinur can be rewritten as follows.

▶ **Theorem 15** (Gap amplification). *Fix a parameter t. Given a Label Cover $G = (V, E, \Pi, \Sigma)$ where Σ is an absolute constant, there is a polynomial time reduction to output a rectangular Label Cover instance $G' = (V', E', \Pi', \Sigma')$ with the alphabet size $|\Sigma'| = c(|\Sigma|, t)$ such that*

- *If G is satisfiable, G' is satisfiable as well.*
- *If every labeling violates at least δ fraction of the constraints of G, then every labeling violates at least $\Omega(\delta\sqrt{t})$ fraction of the constraints of G'.*

Choosing t large enough constant and iterating Theorem 13 and Theorem 15 $\log(m)$ times proves the PCP theorem.

References

1. Sanjeev Arora, Carsten Lund, Rajeev Motwani, Madhu Sudan, and Mario Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998. doi:10.1145/278298.278306.
2. Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998. doi:10.1145/273865.273901.
3. Per Austrin, Venkatesan Guruswami, and Johan Håstad. $(2 + \epsilon)$-Sat is NP-hard. *SIAM J. Comput.*, 46(5):1554–1573, 2017. doi:10.1137/15M1006507.
4. Libor Barto, Andrei Krokhin, and Ross Willard. Polymorphisms, and how to use them. In Andrei Krokhin and Stanislav Živný, editors, *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 1–44. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.1.
5. Mihir Bellare, Oded Goldreich, and Madhu Sudan. Free bits, PCPs, and nonapproximability-towards tight results. *SIAM J. Comput.*, 27(3):804–915, 1998. doi:10.1137/S0097539796302531.
6. Jakub Bulín, Andrei Krokhin, and Jakub Opršal. Algebraic approach to promise constraint satisfaction. In *Proceedings of the 51st Annual ACM SIGACT Symposium on the Theory of Computing (STOC '19)*, New York, NY, USA, 2019. ACM. doi:10.1145/3313276.3316300.
7. Irit Dinur. The PCP theorem by gap amplification. *Journal of the ACM*, 54(3):12, 2007. doi:10.1145/1236457.1236459.
8. Irit Dinur and Omer Reingold. Assignment testers: Towards a combinatorial proof of the PCP theorem. *SIAM J. Comput.*, 36(4):975–1024, 2006.
9. Ehud Friedgut, Gil Kalai, and Assaf Naor. Boolean functions whose Fourier transform is concentrated on the first two levels. *Adv. in Applied Math.*, 29:427–437, 2002.
10. Venkatesan Guruswami and Ryan O'Donnell. The PCP theorem and hardness of approximation: Notes on lectures 7–9. https://courses.cs.washington.edu/courses/cse533/05au/, 2005.
11. Johan Håstad. Some optimal inapproximability results. *Journal of the ACM*, 48(4):798–859, 2001. doi:10.1145/502090.502098.
12. Jaikumar Radhakrishnan and Madhu Sudan. On Dinur's proof of the PCP theorem. *Bull. Amer. Math. Soc.*, 44:19–61, 2007.

A Derandomization of the gadget decoding

In this appendix, we provide a derandomization of the decoding used in Theorem 8. This requires only a little additional argument. The idea is, instead of decoding to i with probability $\hat{f}(i)^2$, to decode to the i with the largest $\hat{f}(i)^2$. We set i_f to be such i. In this light, the reduction can be analyzed by analyzing completeness and soundness of the gadget separately without considering the rest of the instance. The following lemma then shows that the gadget has perfect completeness and soundness 99% not depending on the parameters n and m (the alphabet sizes), π and σ.

▶ **Lemma 16.** *There is a gadget with inputs n, m, k, $\pi\colon [n] \to [k]$, and $\sigma\colon [m] \to [k]$ that produces an instance of* $\mathrm{CSP}(R)$ *with variables $f(a_1, \ldots, a_n)$ and $g(a_1, \ldots, a_m)$ such that*
1. *if $\pi(i) = \sigma(j)$ then p_i and p_j satisfy all the constraints, and*
2. *if at least 99% of the constraints are satisfied, then $\pi(i_f) = \sigma(i_g)$.*

Proof. First, we bound the probability that the test accepts. For the long code test, starting with the first expression in (7), we obtain the following bound.

$$1 - \frac{1}{2}\sum_{|\alpha|>1} \hat{f}(\alpha)^2\,(3/8) - \frac{1}{4}\sum_{\alpha\cap\beta=\emptyset} \hat{f}(\alpha)^2\hat{f}(\beta)^2(1/2)^{|\alpha\cup\beta|} \le 1 - \frac{3}{16}\sum_{|\alpha|>1}\hat{f}(\alpha)^2 - \frac{1}{16}\sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2$$

For the consistency test, we use the bound from Lemma 11. Thus the overall probability that the whole test accepts is at most

$$1 - \frac{1}{16}\sum_{|\alpha|>1}\hat{f}(\alpha)^2 - \frac{1}{48}\sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2 - \frac{1}{16}\sum_{|\alpha|>1}\hat{g}(\alpha)^2 - \frac{1}{48}\sum_{i\neq j}\hat{g}(i)^2\hat{g}(j)^2 - \frac{1}{12}\sum_{i,j:\pi(i)\neq\sigma(j)}\hat{f}(i)^2\hat{g}(j)^2.$$

Given that the acceptance probability is at least $99\% > 1 - 1/96$, we get that

$$\sum_{|\alpha|>1}\hat{f}(\alpha)^2 \le 1/6 \tag{15}$$

$$\sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2 \le 1/2 \tag{16}$$

$$\sum_{|\alpha|>1}\hat{g}(\alpha)^2 \le 1/6 \tag{17}$$

$$\sum_{i\neq j}\hat{g}(i)^2\hat{g}(j)^2 \le 1/2 \tag{18}$$

$$\sum_{i,j:\pi(i)\neq\sigma(j)}\hat{f}(i)^2\hat{g}(j)^2 \le 1/8 \tag{19}$$

From Parseval's identity and (15), we get that $1 \ge \sum_i \hat{f}(i)^2 \ge 5/6$. Recall that i_f is such i that $\hat{f}(i)^2$ is maximal. Then using the above and (16), we obtain that

$$\hat{f}(i_f)^2 \ge \hat{f}(i_f)^2\sum_i \hat{f}(i)^2 \ge \sum_i \hat{f}(i)^4 = \sum_{i,j}\hat{f}(i)^2\hat{f}(j)^2 - \sum_{i\neq j}\hat{f}(i)^2\hat{f}(j)^2 \ge (5/6)^2 - 1/2 = 4/9. \tag{20}$$

Similarly, from (17) and (18), we get $\hat{g}(i_g)^2 \ge 4/9$. Finally, since $\hat{f}(i_f)^2\hat{g}(i_g)^2 \ge (4/9)^2 > 1/8$, we have that $\pi(i_f) = \sigma(i_g)$ otherwise (19) cannot be true. ◀

Theorem 8 can be also directly obtained from this lemma albeit with a worse constant than in the above proof: Let $C = 1\%$ and assume that $\delta < 1$. Given that the resulting CSP instance has an assignment fails no more than $C\delta$-fraction of the constraints, we derive that in at least $(1-\delta)$-fraction of the gadgets, no more than C-fraction of constraints are unsatisfied. Lemma 16 then shows that the assignment $s\colon u \mapsto i_{f_u}$ is an assignment of the Label Cover instance that satisfies all the constraints corresponding to these gadgets. This completes the proof.

L_p Pattern Matching in a Stream

Tatiana Starikovskaya
DIENS, École normale supérieure, PSL Research University, Paris, France
tat.starikovskaya@gmail.com

Michal Svagerka
ETH Zürich, Switzerland
michal.svagerka@alumni.ethz.ch

Przemysław Uznański
Institute of Computer Science, University of Wrocław, Poland
puznanski@cs.uni.wroc.pl

Abstract

We consider the problem of computing distance between a pattern of length n and all n-length subwords of a text in the streaming model.

In the streaming setting, only the Hamming distance (L_0) has been studied. It is known that computing the exact Hamming distance between a pattern and a streaming text requires $\Omega(n)$ space (folklore). Therefore, to develop sublinear-space solutions, one must relax their requirements. One possibility to do so is to compute only the distances bounded by a threshold k, see [SODA'19, Clifford, Kociumaka, Porat] and references therein. The motivation for this variant of this problem is that we are interested in subwords of the text that are similar to the pattern, i.e. in subwords such that the distance between them and the pattern is relatively small.

On the other hand, the main application of the streaming setting is processing large-scale data, such as biological data. Recent advances in hardware technology allow generating such data at a very high speed, but unfortunately, the produced data may contain about 10% of noise [Biol. Direct.'07, Klebanov and Yakovlev]. To analyse such data, it is not sufficient to consider small distances only. A possible workaround for this issue is the $(1 \pm \varepsilon)$-approximation. This line of research was initiated in [ICALP'16, Clifford and Starikovskaya] who gave a $(1 \pm \varepsilon)$-approximation algorithm with space $\widetilde{\mathcal{O}}(\varepsilon^{-5}\sqrt{n})$.

In this work, we show a suite of new streaming algorithms for computing the Hamming, L_1, L_2 and general L_p $(0 < p < 2)$ distances between the pattern and the text. Our results significantly extend over the previous result in this setting. In particular, for the Hamming distance and for the L_p distance when $0 < p \leq 1$ we show a streaming algorithm that uses $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n})$ space for polynomial-size alphabets.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms

Keywords and phrases streaming algorithms, approximate pattern matching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.35

Category APPROX

Funding *Tatiana Starikovskaya*: This work was partially funded by the grant ANR-19-CE48-0016 from the French National Research Agency (ANR).
Przemysław Uznański: Supported by Polish National Science Centre grant 2019/33/B/ST6/00298.

1 Introduction

In the problem of pattern matching, we are given a pattern P of length n and a text T and must find all occurrences of P in T. A particularly relevant variant of this fundamental question is approximate pattern matching, where the goal is to find all subwords of the text that are similar to the pattern. This can be restated in the following way: given a


© Tatiana Starikovskaya, Michal Svagerka, and Przemysław Uznański;
licensed under Creative Commons License CC-BY



pattern P, a text T, and a distance function, compute the distance between P and every n-length subword of T. A very natural similarity measure for words is the Hamming distance. Furthermore, if both P and T are over an integer alphabet Σ, one can consider the Manhattan distance or the Euclidean distance.

▶ **Definition 1** (Hamming, Manhattan and Euclidean distances). *For a vector $U = u_1u_2 \dots u_n$, its Hamming norm is defined as $\|U\|_H = |\{i : u_i \neq 0\}|$, Manhattan norm is defined as $\|U\|_1 = \sum_i |u_i|$ and Euclidean norm is defined as $\|U\|_2 = \left(\sum_i u_i^2\right)^{1/2}$. For two words $V = v_1v_2 \dots v_n$ and $W = w_1w_2 \dots w_n$, their Hamming distance is defined as $\|V - W\|_H$, their Manhattan distance as $\|V - W\|_1$, and their Euclidean distance as $\|V - W\|_2$.*

Those distance functions naturally generalize to the so called L_p distances, where $p > 0$ is the exponent.

▶ **Definition 2** (p'th moment, p'th norm). *For a vector $U = u_1u_2 \dots u_n$ and $p \geq 0$, its p'th moment F_p is defined as $F_p(U) = \sum_i |u_i|^p$, and for $p > 0$ its L_p norm is defined as $\|U\|_p = F_p(U)^{1/p} = (\sum_i |u_i|^p)^{1/p}$. For two words $V = v_1v_2 \dots v_n$ and $W = w_1w_2 \dots w_n$ considered as vectors, the p'th moment of their difference is $F_p(V-W)$ and their L_p distance is defined as $\|V - W\|_p = F_p(V - W)^{1/p} = (\sum_i |v_i - w_i|^p)^{1/p}$.*

In other words, the Manhattan distance is the L_1 distance, the Euclidean distance is the L_2 distance, and the Hamming distance can be considered as the L_0 distance.

Below we assume that the length of the text is $2n$, as any algorithm on a text of larger length can be reduced to repeated application of an algorithm that runs on texts of length $2n$. This is done by splitting the text into blocks of length $2n$ which overlap by n characters.

Offline setting. For the Hamming distance, the problem has been extensively studied in the offline setting, where we assume random access to the input. The first algorithm, for a constant-size alphabet, was shown by Fischer and Paterson [21]. The algorithm uses $\mathcal{O}(n \log n)$ time and in substance computes the Boolean convolution of two vectors a constant number of times. This was later extended to polynomial-size alphabets in [1, 33]. With a somewhat similar approach, the same complexity can be achieved for the L_1 distance in [13]. Later, in [34, 35] the authors proved that these problems must have equal (up to polylogarithmic factors) complexities by showing reductions from the Hamming to the L_1 distance and back.

To improve the complexity for large alphabets, the natural next step was to study approximation algorithms. Until very recently, the fastest $(1 \pm \varepsilon)$-approximation algorithm for computing the Hamming distances was by Karloff [29]. The algorithm combines random projections from an arbitrary alphabet to the binary one and Boolean convolution to solve the problem in $\mathcal{O}(\varepsilon^{-2}n \log^3 n)$ time. In a breakthrough paper Kopelowitz and Porat [31] gave a new approximation algorithm improving the time complexity to $\mathcal{O}(\varepsilon^{-1}n \log^3 n \log \varepsilon^{-1})$, which was later significantly simplified [32]. Using a similar technique, Gawrychowski and Uznański [23] showed an approximation algorithm for computing the L_1 distance in $\mathcal{O}(\varepsilon^{-1}n \log^4 n)$ (randomized) time, later made deterministic in time $\mathcal{O}(\varepsilon^{-1}n \log^2 n)$ in [39]. Using similar techniques, the authors of [39] gave $\widetilde{\mathcal{O}}(\varepsilon^{-1}n)$-time $(1 + \varepsilon)$-approximation algorithm for L_p distances for any constant positive p.[1]

[1] Across the paper we use $\widetilde{\mathcal{O}}$ to indicate that we are suppressing poly-log(n) factors.

Streaming setting. In the streaming setting, we assume that the pattern and the text arrive as streams, one character at a time (the pattern arrives before the text). The main objective is to design algorithms that use as little space as possible, and we must account for all the space used by the algorithm, including the space required to store the input, in full or in part. It is also often the case that the text arrives at a very high speed and we must be able to process it faster than it arrives to fulfil the space guarantees, preferably, in real time. To this aim, the time complexity of streaming algorithms is defined as the worst-case amount of time spent on processing one character of the text, i.e. *per arrival*.

In the streaming setting, only the Hamming distance (L_0) has been studied. It is known that computing the Hamming distance between a pattern and a streaming text exactly requires $\Omega(n)$ space, even for the binary alphabet and with a small probability error allowed, which can be shown by a straightforward reduction to communication complexity (folklore).

Therefore, to develop sublinear-space solutions, one must relax their requirements. One possibility to do so is to compute only the distances bounded by a threshold k. This variant of the problem is often reffered to as *k-mismatch problem*. The k-mismatch problem has been extensively studied in the literature [15, 16, 25, 38], with this line of work reaching $\widetilde{\mathcal{O}}(k)$ memory complexity and $\widetilde{\mathcal{O}}(\sqrt{k})$ time per input character. The motivation for this variant of this problem is that we are interested in subwords of the text that are similar to the pattern, in other words, the distance between the pattern and the text should be relatively small. On the other hand, the main application of the streaming setting is processing large-scale data, such as biological data. To decrease the cost of generating such data, recently new hardware approaches have been developed. They have become widely used due to cost efficiency, but unfortunately, the produced data may contain about 10% of noise [30]. To analyse such data, it is not sufficient to consider small distances only, and a possible workaround for this issue is $(1 \pm \varepsilon)$-approximation. This line of research was initiated by Clifford and Starikovskaya [17] who gave a $(1 \pm \varepsilon)$-approximation algorithm with space $\widetilde{\mathcal{O}}(\varepsilon^{-5}\sqrt{n})$ that uses $\widetilde{\mathcal{O}}(\varepsilon^{-4})$ time per arriving character of the text.

Independently and in parallel with this work, authors of [12] showed a $(1\pm\varepsilon)$-approximation streaming algorithm for the k-mismatch problem that uses $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{k})$ space. For a special case of $k = n$, they show how to reduce the space further to $\widetilde{\mathcal{O}}(\varepsilon^{-1.5}\sqrt{n})$. Compared to our solution, their algorithm has worse time complexity of $\widetilde{\mathcal{O}}(\varepsilon^{-3})$ per arrival, and more importantly, it is not obvious whether it can be generalised to other L_p norms as it uses a very different set of techniques.

Sliding window. The problem of computing distance between P and every n-length subword of T in the streaming setting resembles the problem of maintaining the L_p norm of a n-length suffix of a streaming text, also referred to as *sliding window*. In fact, the latter is a simplification of the former, with setting $P = [0, 0, \ldots, 0]$. There is an extensive line of work on maintaining the L_p norm of a sliding window, refer to [4, 5, 6, 7, 8, 18] and references therein. The main message is that the norm of a sliding window can be maintained efficiently, e.g. for $1 \leq p \leq 2$ the L_p norms can be maintained $(1 \pm \varepsilon)$-approximately in space $\widetilde{\mathcal{O}}(\varepsilon^{-1})$. However, those results do not translate to our case: in the sliding window, one can easily isolate "heavy hitters", that is updates with a significant contribution to the output. In our case, the contribution of an update depends on its relative position to the pattern, and one can easily construct instances where a contribution of a position in the text changes drastically relative to its alignment with the pattern, which necessitates a significantly different approach.

1.1 Our results

In this work, we show a suite of new streaming algorithms for computing the Hamming, L_1, L_2 and general L_p ($0 < p \leq 2$) distances between the pattern and the text. Our results significantly improve and extend the results of [17].

▶ **Theorem 3.** *Given a pattern P of length n and a text T over an alphabet $\Sigma = [1, 2, \ldots, \sigma]$, where $\sigma = n^{\mathcal{O}(1)}$, there is a streaming algorithm that computes a $(1 \pm \varepsilon)$-approximation of the L_p distance between P and every n-length subword of T correctly w.h.p.*

1) *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log \sigma)$ space, and $\widetilde{\mathcal{O}}(\varepsilon^{-2})$ time per arrival when $p = 0$ (Hamming distance);*
2) *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log^2 \sigma)$ space and $\widetilde{\mathcal{O}}(\sqrt{n} \log \sigma)$ time per arrival when $p = 1$ (Manhattan distance);*
3) *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log^2 \sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n})$ time per arrival when $0 < p < 1/2$;*
4) *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log^2 \sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-3}\sqrt{n})$ time per arrival when $p = 1/2$;*
5) *in $\widetilde{\mathcal{O}}(\varepsilon^{-2}\sqrt{n} + \log^2 \sigma)$ space and $\widetilde{\mathcal{O}}(\sigma^{\frac{2p-1}{1-p}}\sqrt{n}/\varepsilon^{2+3\cdot\frac{2p-1}{1-p}})$ time per arrival when $1/2 < p < 1$;*
6) *in $\widetilde{\mathcal{O}}(\varepsilon^{-2-p/2}\sqrt{n}\log^2 \sigma)$ space and $\mathcal{O}(\varepsilon^{-p/2}\sqrt{n} + \varepsilon^{-2}\log \sigma)$ time per arrival for $1 < p \leq 2$.*

We also improve and extend the space lower bound of [17], who showed that any streaming algorithm that computes a $(1 \pm \varepsilon)$-approximation of the Hamming distance between a pattern and a streaming text must use $\Omega(\varepsilon^{-2} \log^2 n)$ bits for all ε such that $1/\varepsilon < n^{1/2-\gamma}$ for some constant γ (condition inherited from [27]). We show the following result:

▶ **Lemma 4.** *Let $2 \leq 1/\varepsilon < n$ and $0 \leq p \leq 2$. Any $(1 \pm \varepsilon)$-approximation algorithm that computes the L_p distance between a pattern and a streaming text for each alignment, must use $\Omega(\min(1/\varepsilon^2, n))$ bits of space.*

1.2 Techniques

At a very high level, the structure of all algorithms presented in this paper is similar to that of [17]. We process the text by blocks of length $b \approx \sqrt{n}$. To compute an approximation of the distance / the p'th moment at a particular alignment, we divide the pattern into two parts: a prefix of length $\leq b$ aligned with a suffix of some block of the text, and the remaining suffix (see Fig. 1). We compute an approximation of the distance / the p'th moment for both of the parts and sum them up to obtain the final answer. Our main contribution is a set of new tools that allows computing the approximations efficiently.

To be able to compute the approximation of the distance / the p'th moment between the prefix and the corresponding block of the text, we compute, while reading each block of the text, its compact lossy description that we refer to as *prefix encoding*. The prefix encoding captures the relation between the read block and the prefix of the pattern of length b. To compute the distance / the p'th moment between the suffix and the text, we will use *suffix sketches*. For each position i of the text, the suffix sketch describes the subword $T[b \cdot k + 1, i]$ of the text where k is the smallest integer such that $i - b \cdot k \leq n$ (see Fig. 1).

For the Hamming distance, we define the prefix encodings in Section 2.1 and the suffix sketches in Section 3.1. Our Hamming prefix encoding introduces a novel use of a known technique called *subsampling*. The prefix encodings are used to approximate the distance between any suffix of one word and the prefix of another word of the same length. In brief, the idea is to replace each character of the two words by the don't care character "?", a special character that matches any other character of the alphabet. We repeat the process a logarithmic number of times to create a logarithmic number of pairs of "subsamples". For each pair, we find the longest suffix of one subsample that matches the prefix of the

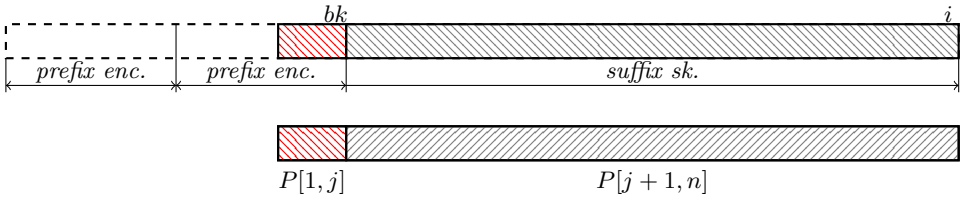


Figure 1 High level structure of the algorithms. To compute the distance between the prefix (red) of the pattern and the text, we use the prefix encoding, between the suffix (grey) and the text we use the suffix sketch.

second subsample up to at most $\Theta(1/\varepsilon^2)$ mismatches. We then show that this information can be used to approximate the Hamming distance between any suffix-prefix pair. Similar techniques were used in [3, 19, 22, 24, 28, 37] for estimating the Hamming norm in streams. The crucial difference with our approach is that we must be able to compute the Hamming norm of any suffix-prefix pair of the two words, and we must be able to do it efficiently. As for the suffix sketches, for the binary alphabet we use the sketches introduced in [17]. We then show a reduction from arbitrary alphabets to the binary alphabet, which improves the space consumption of Hamming suffix sketches by a factor of $1/\varepsilon^2$.

We can solve the problem of L_1 (Manhattan distance) pattern matching by replacing each character of the pattern and of the stream with its unary encoding and running the solution for the Hamming distance. However, this would introduce a multiplicative factor of σ (the size of the alphabet) to the time complexity. We show efficient randomised reductions from the Manhattan to Hamming distance that allow simulating the solution for the Hamming distance without a significant overhead. In particular, to design the prefix encodings we use random shifting and rounding, while for the suffix sketches we use range-summable hash functions [9]. We show the Manhattan prefix encodings in Section 2.2 and the Manhattan suffix sketches in Section 3.2.

For generic L_p distances, $0 < p \leq 2$, we discuss the prefix encodings in Section 2.4 and the suffix sketches in Section 3.3. Our approach to L_p prefix encodings is rather involved. In the case of $0 < p < 1$, we construct a novel embedding from L_p^p space into the Hamming space, which might be of independent interest. While the target dimension of the Hamming space is large, we construct the embedding in such a way that each value is mapped into a compressible sequence of form $c_1^{d_1} \dots c_t^{d_t}$ for some small value of t, and where values of $d_1, \dots, d_t$ are constant across all input values. Such compressed representation allows us to efficiently apply the subsampling framework and reduce the problem to the Hamming distance case. For $1 < p \leq 2$, we identify a logarithmic number of anchor suffixes, and partition each of them into ε^{-p} words of roughly even contribution to the distance. We then use the partition to decode prefix-suffix distance queries for arbitrary length queries. Such construction is a generalization and improvement of the approach presented in [17]. For suffix sketches, we simply use the p-stable distributions [26].

Finally, we combine the prefix encodings and the suffix sketches to prove Theorem 3 in Section 4. To simplify the notation, we use $x \overset{\varepsilon}{=} y$ to denote $(1-\varepsilon)y \leq x \leq (1+\varepsilon)y$ from now on. We will also use the fact that for $p > 0$ we can speak of approximating the p'th moment of differences between the pattern and the n-length substrings of the text and the L_p distances between the pattern and the n-length substrings of the text interchangeably, it changes the complexities up to a constant factor only:

▶ **Observation 5.** *For any constant $p > 0$ and $\varepsilon < 1/2$, there is a constant C_p such that finding a $(1 \pm C_p \cdot p\varepsilon)$ approximation of the p'th moment of a vector suffices for $(1 \pm \varepsilon)$-approximating its p'th norm, and finding a $(1 \pm C_p \cdot \varepsilon/p)$ approximation of its p'th norm suffices for $(1 \pm \varepsilon)$-approximating its p'th moment.*

2 Prefix encodings

In this section we present a solution to the following problem. Imagine we have a block of text $T'[1, b] = T[i + 1, i + b]$ and a prefix of the pattern $P' = P[1, b]$. We want to find a compressed representation (encoding) of T' so that the following is possible: given any $1 \le d \le b$, the compressed representation of T', and P' (explicitly), we can $1 \pm \varepsilon$ approximate $\|T'' - P''\|_p$, where $T'' = T'[b - d + 1, b]$ is a suffix of T' and $P'' = P'[1, d]$ is a prefix of P'.

We start by presenting a solution to the Hamming distance case, which is a basis to our solution for all other L_p norms for $0 < p \le 2$.

2.1 Hamming (L_0) distance

Recall that "?" is the don't care character, a special character that matches any other character of the alphabet.

▶ **Definition 6** (Hamming subsampling). *Consider a word U of length n. Let $q = \lceil 3 \log n \rceil$ and let $h(i) : [n] \to \{0, 1\}^q$ be a function drawn at random from a pairwise independent family. For $r = 0, \ldots, q$, we define the r-th level Hamming subsample of U, $\mathsf{hSub}_r(U)$, as follows:*

$$\mathsf{hSub}_r(U)[i] = \begin{cases} U[i], & \text{if the } r \text{ lowest bits of } h(i) \text{ are all } 0; \\ ?, & \text{otherwise.} \end{cases}.$$

In particular, $\mathsf{hSub}_0(U) = U$.

Fix an integer $k = \Theta(1/\varepsilon^2)$ large enough. For two words U, V, consider the following estimation procedure:

▶ **Algorithm 7.**

1. *Denote X_r to be the Hamming distance between $\mathsf{hSub}_r(U)$ and V and let $f = \min\{i : X_i \le k\}$.*[2]
2. *Output $Z_f = 2^f \cdot X_f$ as an estimate of $\|U - V\|_H$.*

The following lemma is a rephrasing of a known result regarding subsampling in estimation of the Hamming norm (cf. [3, Theorem 3], or [24, Theorem 2]).

▶ **Lemma 8.** *For Z_f as in Algorihtm 7 there is $Z_f \overset{\varepsilon}{=} \|U - V\|_H$ with probability at least $3/4$.*

Since the subsampling is performed independently for each position, one can use subsampling to approximate the Hamming distance between any suffix of B and any prefix of P of equal lengths in a similar fashion.

We are now ready to define the Hamming prefix encoding of a block. For brevity, let $B_r^j = \mathsf{hSub}_r(B)[b - j + 1, b]$ and $P_r^j = P[1, j]$ (the same for all r). Furthermore, given two words U, V of equal length, define the *mismatch information* $\mathsf{MI}(U, V) = \{(i, U[i], V[i]) : U[i] \text{ does not match } V[i]\}$.

[2] We emphasize that $\mathsf{hSub}_r(U)$ contains don't care characters, so the Hamming distance is defined as the number of pairs of characters of $\mathsf{hSub}_r(U)$ and V that do not match.

▶ **Definition 9.** *Consider a b-length block B of the text T. For each $0 \le r \le \lceil 3 \log n \rceil$, let $j^*(r)$ be the maximal integer such that the Hamming distance between $B_r^{j^*(r)}$ and $P_r^{j^*(r)}$ is at most $k = \Theta(\varepsilon^{-2})$. We define the Hamming prefix encoding of B to be a tuple of pairs $j^*(r), \mathsf{MI}(B_r^{j^*(r)}, P_r^{j^*(r)})$.*

Note that the prefix encoding of B uses $\mathcal{O}(k \log n) = \mathcal{O}(\varepsilon^{-2} \log n)$ space. We can compute it efficiently:

▶ **Lemma 10.** *Assume constant-time random access to $P[1, b]$. Given a b-length block B of the text T, its Hamming prefix encoding can be computed in $\widetilde{\mathcal{O}}(kb) = \widetilde{\mathcal{O}}(b\varepsilon^{-2})$ time.*

Proof. To compute the encoding, we use the algorithm of [14]. Formally, for each r we create a word T' by appending b don't care characters to the subsample hSub_r. The algorithm of [14] can be used to find all b-length subwords of T' that match $P[1, b]$ with up to k mismatches, moreover for each of these subwords the algorithm outputs the mismatch information. We take the leftmost subword only, which corresponds to $j^*(r)$ because of the don't care characters. In total, our algorithm uses $\widetilde{\mathcal{O}}(kb) = \widetilde{\mathcal{O}}(\varepsilon^{-2}b)$ time. ◀

We now show how to compute the Hamming distance between any j-length suffix of B and any j-length prefix of P given $P[1, b]$ and the Hamming prefix encoding of a block B.

▶ **Lemma 11.** *Given the prefix encoding of a b-length block B of the text T, there is an algorithm that computes, for any $j = 1, \ldots, b$, a $(1 + \varepsilon)$-approximation of the Hamming distance between the j-length suffix of B and the j-length prefix of P in $\widetilde{\mathcal{O}}(kb) = \widetilde{\mathcal{O}}(b\varepsilon^{-2})$ time.*

Proof. Denote X_r to be the Hamming distance between P_r^j and B_r^j. We compute the smallest f such that $X_f \le k$ in the following way. For each r, we use $\mathsf{MI}(B_r^{j^*(r)}, P_r^{j^*(r)})$ to restore $B_r^{j^*(r)}$. We then append $P_r^{j^*(r)}$ with b don't care characters and run the algorithm of [14] for the resulting text and the pattern. This allows to compute X_r for all $j \le j^*(r)$, and if $j > j^*(r)$, then $X_f > k$ by definition. In total, the algorithm takes $\widetilde{\mathcal{O}}(kb) = \widetilde{\mathcal{O}}(\varepsilon^{-2}b)$ time. ◀

2.2 Manhattan (L_1) distance

Recall a word morphism $\nu : \Sigma \to \{0,1\}^\sigma$, $\nu(a) = 1^a 0^{\sigma - a}$. Our goal in this section is to simulate implicitly procedures from Lemma 10 and Lemma 11 on words $\nu(B)$ and $\nu(T)$ without introducing any significant overhead.

▶ **Definition 12** (Manhattan scaling)**.** *Consider a word U of length n. Let $q = \lceil 3 \log n\sigma \rceil$ and let $h : [n] \to 2^q$ be a function drawn at random from a 4-wise independent family. For $r = 0, \ldots, q$, we define the r-th level Manhattan subsample of U, $\mathsf{mSub}_r(U)$, as a word of length n such that $\mathsf{mSub}_r(U)[i] = \left\lfloor \frac{U[i] + (h(i) \bmod 2^r)}{2^r} \right\rfloor$. In particular, $\mathsf{mSub}_0(U) = U$.*

Fix an integer $k = \Theta(1/\varepsilon^2)$ large enough. For words U, V, consider $\mathsf{mSub}_r(U), \mathsf{mSub}_r(V)$ for all $r = 0, \ldots, q$, and the following estimation procedure:

▶ **Algorithm 13.**
1. *Denote $X_r = \|\mathsf{mSub}_r(U) - \mathsf{mSub}_r(V)\|_1$ and let $f = \min\{i : X_i \le k\}$.*
2. *Output $Z_f = 2^f \cdot X_f$ as an estimate of $\|U - V\|_1$.*

▶ **Lemma 14.** *For Z_f as in Algorihtm 13 there is $Z_f \overset{\varepsilon}{=} \|U - V\|_1$ with probability $\ge 3/4$.*

To approximate the Manhattan distance between any suffix of B and any prefix of P of equal lengths, we define the encoding similar to the Hamming distance case. Specifically, we still use the *mismatch information*, building on the fact that for any two words $\|U - V\|_H \leq \|U - V\|_1$ and from the mismatch information the exact value of $\|U - V\|_1$ can be found. We define $B_r^j = \mathsf{mSub}_r(B)[b - j + 1, b]$ as before, but change the definition of P_r^j slightly. Intuitively, we define P_r^j to be the j-length prefix of P subsampled in a synchronized way with B_r^j. Formally, $P_r^j[i] = \left\lfloor \frac{P[i] + (h(b-j+i) \bmod 2^r)}{2^r} \right\rfloor$.

▶ **Definition 15.** *Consider a b-length block B of the text T. For each $0 \leq r \leq \lceil 3 \log n\sigma \rceil$, let $j^*(r)$ be the maximal integer such that the Manhattan distance between $B_r^{j^*(r)}$ and $P_r^{j^*(r)}$ is at most $k = \Theta(\varepsilon^{-2})$. We define the Manhattan prefix encoding of B to be a tuple of pairs $j^*(r), \mathsf{MI}(B_r^{j^*(r)}, P_r^{j^*(r)})$.*

Note that the prefix encoding of B uses $\mathcal{O}(k \log n\sigma) = \mathcal{O}(\varepsilon^{-2} \log n)$ space.

▶ **Lemma 16.** *Assume constant-time random access to $P[1, b]$. Given a b-length block B of the text T, its Manhattan prefix encoding can be computed in $\widetilde{\mathcal{O}}(b^2)$ time and $\widetilde{\mathcal{O}}(b)$ space.*

Proof. Let $q = \lceil 3 \log n\sigma \rceil$. For each $r = 0, \ldots, q$ and $j = 1, \ldots, b$ we compare B_r^j and P_r^j character by character in $\mathcal{O}(b)$ time to find $j^*(r)$ and the corresponding mismatch information. The claim follows. ◀

▶ **Lemma 17.** *Given the prefix encoding of a b-length block B of the text T, there is an algorithm that computes, for all $j = 1, \ldots, b$, a $(1 \pm \varepsilon)$-approximation of the Manhattan distance between the j-length suffix of B and the j-length prefix of P in $\widetilde{\mathcal{O}}(b^2)$ time.*

Proof. Denote $X_r = \|P_r^j - B_r^j\|_H$. We compute the smallest f such that $X_f \leq k$ in the following way. For each r, we use $\mathsf{MI}(B_r^{j^*(r)}, P_r^{j^*(r)})$ to restore $B_r^{j^*(r)}$. If $j > j^*(r)$, the Manhattan distance between P_r^j and B_r^j is at least k. Otherwise, we compare P_r^j and B_r^j character by character to compute the Manhattan distance in $\mathcal{O}(b)$ time. The claim follows. ◀

2.3 Generic (L_p) distance for $0 < p < 1$

Our goal is to construct a morphism (parametrised by p) acting as a randomized embedding of $(L_p)^p$ into the Hamming distance. The intuition behind our approach is as follows. Let $r_0, r_1, \ldots \in [0, 1]$ be a sequence of real numbers picked independently and u.a.r. Define a sequence of values

$$d_i = \begin{cases} \varepsilon^{-1} \cdot (1+\varepsilon)^{pi} & \text{when } i > 0 \\ \varepsilon^{-1} \cdot \frac{(1+\varepsilon)^p}{(1+\varepsilon)^p - 1} & \text{when } i = 0 \end{cases}$$

and for a character $c \in \Sigma$ consider sequence of characters $c_0, c_1, \ldots$ where $c_i = \lfloor \frac{c}{(1+\varepsilon)^i} + r_i \rfloor$ (similarly, a character c' defines a sequence $c'_0, c'_1, \ldots$). Now consider two characters $c, c' \in \Sigma$ such that $|c - c'| = (1+\varepsilon)^\ell$ for some integer ℓ and a random variable $x = \sum_{i=0}^{\infty} d_i \cdot \|c_i - c'_i\|_H$.

There is

$$\begin{aligned}\mathbb{E}[x] &= \sum_{i=0}^{\infty} d_i \cdot \Pr[c_i \neq c'_i] = \sum_{i=0}^{\ell} d_i \cdot 1 + \sum_{i=\ell+1}^{\infty} d_i \cdot \frac{|c-c'|}{(1+\varepsilon)^i} = \\ &= \varepsilon^{-1}\frac{(1+\varepsilon)^p}{(1+\varepsilon)^p-1} + \varepsilon^{-1}\sum_{i=1}^{\ell}\left((1+\varepsilon)^p\right)^i + \varepsilon^{-1}|c-c'|\sum_{i=\ell+1}^{\infty}\left((1+\varepsilon)^{p-1}\right)^i = \\ &= \varepsilon^{-1}\frac{((1+\varepsilon)^{\ell+1})^p}{(1+\varepsilon)^p-1} + |c-c'|\varepsilon^{-1}\frac{((1+\varepsilon)^{\ell+1})^{p-1}}{1-(1+\varepsilon)^{p-1}} = \\ &= |c-c'|^p\left(\frac{(1+\varepsilon)^p}{(1+\varepsilon)^p-1} + \frac{1}{(1+\varepsilon)^{1-p}-1}\right)\varepsilon^{-1} \approx \varepsilon^{-2}|c-c'|^p\frac{1}{p(1-p)}. \end{aligned} \tag{1}$$

We thus see that an idealized morphism of the form $\varphi : c \to c_0^{d_0}c_1^{d_1}\dots$ would have the property that $\|U-V\|_p^p \sim \|\varphi(U)-\varphi(V)\|_H$ on words of length n. But there are the following issues: (i) characters are mapped into infinite length words, (ii) number of repetitions of characters (d_i) is fractional, (iii) we cannot guarantee that character distance is always of form $(1+\varepsilon)^i$ and (iv) the distance is preserved only in expectation. We show how to overcome these issues to achieve the following result:

▶ **Theorem 18.** *Given $0 < p < 1$ and $\varepsilon > 0$ there is a word morphism $\varphi : c \in \Sigma \to c_0^{d_0}c_2^{d_2}\dots c_{t-1}^{d_{t-1}}$ such that:*

1) *$t = \widetilde{\mathcal{O}}(\varepsilon^{-2})$ when $0 < p < 1/2$, $t = \widetilde{\mathcal{O}}(\varepsilon^{-3})$ when $p = 1/2$ and $t = \widetilde{\mathcal{O}}(\sigma^{\frac{2p-1}{1-p}}/\varepsilon^{2+3\cdot\frac{2p-1}{1-p}})$ when $1/2 < p < 1$.*
2) *values of t and $d_0, \dots, d_{t-1}$ do not depend on c,*
3) *there exists a constant $\alpha = \alpha(p,\varepsilon)$ such that for any two words U, V of length at most n, we have $\|U-V\|_p^p \overset{\varepsilon}{=} \alpha \cdot \|\varphi(U)-\varphi(V)\|_H$ with probability at least $9/10$,*
4) *it is enough for the randomness to be realized by a hash function $r : [t] \to [D]$ from a 4-independent hash function family for some $D = \mathrm{poly}(n\sigma\varepsilon^{-1})$, which can be generated from a $\widetilde{\mathcal{O}}(\log\sigma)$ bits size seed.*

We now describe how to use the morphism φ to approximate the L_p distances in a small space. To design an efficient algorithm, we take advantage of the fact that $\varphi(U)$ has a compressed representation of size comparable with the length of U (at least when $p \le 1/2$).

▶ **Definition 19** (L_p scaling)**.** *Consider a word $S = s_1^{e_1}s_2^{e_2}\dots s_m^{e_m}$ of length $m' = \sum_i e_i$. Let $h : [m] \to 2^q$ be a function drawn at random from a 4-wise independent family, where $q = \lceil 3\log m'\rceil$. For $r = 0, \dots, q$, we define the r-th level subsample of S,*

$$\mathsf{Sub}_r(S) = (s_1)^{\left\lfloor \frac{e_1+(h(1) \bmod 2^r)}{2^r}\right\rfloor}(s_2)^{\left\lfloor \frac{e_2+(h(2) \bmod 2^r)}{2^r}\right\rfloor}\dots(s_m)^{\left\lfloor \frac{e_m+(h(m) \bmod 2^r)}{2^r}\right\rfloor}$$

In particular, $\mathsf{Sub}_0(U) = U$.

Consider two words S, Q of form $S = s_1^{e_1}\dots s_m^{e_m}$ and $Q = q_1^{e_1}\dots q_m^{e_n}$. Fix an integer $k = \Theta(1/\varepsilon^2)$ large enough and consider $\mathsf{Sub}_r(S), \mathsf{Sub}_r(Q)$ for all $r = 0, 1, \dots, \lceil 3\log m'\rceil$, where $m' = \sum_i e_i$.

▶ **Algorithm 20.**

1. *Denote $X_r = \|\mathsf{Sub}_r(S) - \mathsf{Sub}_r(Q)\|_H$ and let $f = \min\{i : X_i \le k\}$.*
2. *Output $Z_f = 2^f \cdot X_f$ as an estimate of $\|S-Q\|_H$.*

▶ **Lemma 21.** *For Z_f as in Algorihtm 20 there is $Z_f \overset{\varepsilon}{=} \|S-Q\|_H$ with probability $\ge 3/4$.*

We are now ready to define L_p prefix encodings. Consider a b-length block B of the text and define $B_r^j = \mathsf{Sub}_r(\varphi(B))[(b-j)t+1, bt]$ (t is defined as in Theorem 18). Also, define P_r^j to be the (tj)-length prefix of $\varphi(P)$ subsampled in a synchronized way with B_r^j.

▶ **Definition 22.** *Consider a b-length block B of the text T. For each $r = 0, \ldots, \lceil 3 \log n' \rceil$, where $n' = |\varphi(B)|$, let $j^*(r)$ be the maximal integer such that the Hamming distance between $B_r^{j^*(r)}$ and $P_r^{j^*(r)}$ is at most $k = \Theta(\varepsilon^{-2})$. We define the L_p prefix encoding of B to be a tuple of pairs $j^*(r), \mathsf{MI}(B_r^{j^*(r)}, P_r^{j^*(r)})$.*

The L_p prefix encoding of B uses $\mathcal{O}(k \log n') = \mathcal{O}(\varepsilon^{-2} \log(n\sigma\varepsilon^{-1}))$ space.

▶ **Lemma 23.** *Assume constant-time random access to $P[1, b]$. Given a b-length block B of the text T, its L_p prefix encoding can be computed in $\mathcal{O}(b^2 \cdot t \log n\sigma\varepsilon^{-1})$ time and $\mathcal{O}(b + \varepsilon^{-2} \log n\sigma\varepsilon^{-1})$ space.*

Proof. For each $r = 0, \ldots, \lceil 3 \log n' \rceil$ and $j = 1, \ldots, b$, we compute the Hamming distance between B_r^j and P_r^j in $\mathcal{O}(bt)$ time using the compressed representation to find $j^*(r)$ and the corresponding mismatch information. The claim follows. ◀

▶ **Lemma 24.** *Given the L_p prefix encoding of a b-length block B of the text T, there is an algorithm that computes, for all $j = 1, \ldots, b$, a $(1 \pm \varepsilon)$-approximation of the L_p distance between the j-length suffix of B and the j-length prefix of P in $\widetilde{\mathcal{O}}(b^2 \cdot t \log n\sigma\varepsilon^{-1})$ time and $\mathcal{O}(b + \varepsilon^{-2} \log n\sigma\varepsilon^{-1})$ space.*

Proof. Denote $X_r = \|P_r^j - B_r^j\|_H$. We compute the smallest f such that $X_f \le k$ in the following way. For each r, we use $\mathsf{MI}(B_r^{j^*(r)}, P_r^{j^*(r)})$ to restore $B_r^{j^*(r)}$. If $j > j^*(r)$, the Hamming distance between P_r^j and B_r^j is at least k. Otherwise, we compare P_r^j and B_r^j to compute the Hamming distance in $\mathcal{O}(bt)$ time. The claim follows. ◀

2.4 Generic (L_p) distance for $1 < p \le 2$

For $1 < p \le 2$, we use a scheme similar to the one developped in [17] for the Hamming distance, but adapt it to generic L_p distances. Particularly, we plug in a standard tool used in this situation, the p-stable distribution. We additionally have to adapt the scheme a bit, taking into account that L_p norm is sub-additive under concatenation when $p > 1$.

▶ **Definition 25** (p-stable distribution [40])**.** *For a parameter $p > 0$, we say that a distribution $\mathcal{D}$ is p-*stable *if for all $a, b \in \mathbb{R}$ and random variables X, Y drawn independently from $\mathcal{D}$, the variable $aX + bY$ is distributed as $(|a|^p + |b|^p)^{1/p} Z$, where Z is a random variable with distribution $\mathcal{D}$.*

Consider a word $X = x_1 x_2 \ldots x_n$, and let $\alpha_1, \alpha_2, \ldots, \alpha_n$ be independent random variables drawn from a p-stable distribution $\mathcal{D}$ with expected value $\mu_{\mathcal{D}}$. By Definition 25, we have $\mathbb{E}\left[\sum_i \alpha_i x_i\right] / \mu_{\mathcal{D}} = \|X\|_p$. The p-stable distributions exist for all $0 < p \le 2$, and a random variable X from a p-stable distribution can be generated using the formula $X = \frac{\sin(p\Theta)}{\cos^{1/p}(\Theta)} \left(\frac{\cos(\Theta(1-p))}{\ln(1/r)}\right)^{(1-p)/p}$ [11, 40], where Θ is uniform on $[-\pi/2, \pi/2]$ and r is uniform on $[0, 1]$.

However, to be able to design an efficient sketching scheme that allows to approximate the L_p norm with high probability, there are three technicalities to be overcome: First, one must show that $\sum_i \alpha_i x_i$ concentrates well, second, the formula above assumes infinite precision of computation, and finally, one cannot use fully independent random variables α_i as above

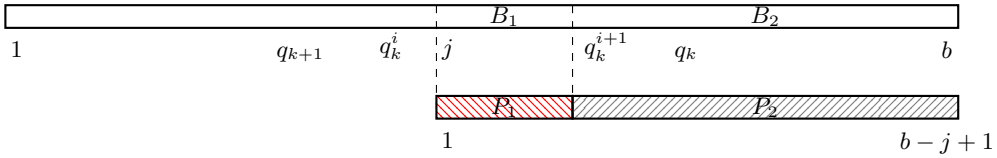


Figure 2 Using the prefix encoding of B to compute the L_p distance between a suffix of B and a prefix of the pattern. To compute the distance between B_1 and P_1, we replace B_1 with a subword of the pattern, and between B_2 and P_2 we use the sketches.

as this would require much space. To overcome these issues, Indyk [26] combined p-stable distributions and pseudorandom generators for bounded space computation [36]. We restate the final result of Indyk below, in the form that will be convenient for us later.

▶ **Theorem 26** (cf. Theorem 2, Theorem 4 [26]). *For any $0 < p \leq 2$, there is a non-uniform streaming algorithm that maintains a sketch $\mathsf{Sketch_p}(S)$ of a word S of length n over an alphabet of size σ such that:*

1) *when a new character of S arrives, the sketch can be updated in $\mathcal{O}(\varepsilon^{-2}\log(n/\varepsilon))$ time;*

2) *the algorithm and the sketch use $\mathcal{O}(\varepsilon^{-2}\log(\sigma n/\varepsilon)\log(n/\varepsilon))$ bits of space.*

Given the sketches $\mathsf{Sketch_p}(X), \mathsf{Sketch_p}(Y)$ of two words X, Y of length n, one can estimate $\|X-Y\|_p$ up to a factor $1\pm\varepsilon$ with probability at least $9/10$ in time $\widetilde{\mathcal{O}}(1/\varepsilon^2)$.

We now proceed to building the L_p prefix encoding by using $\mathsf{Sketch_p}$ and the landmarking technique.

▶ **Definition 27** (L_p prefix encoding). *Let $1 < p \leq 2$. Consider a word S of length b on the alphabet of size σ. Define $q_0 = b$. For $k = 0, \ldots, \lceil \log b\sigma^p \rceil$, let $q_k \leq q_{k-1}$ be the leftmost position such that the p'th moment of the difference between $S[q_k, b]$ and $P[1, b-q_k+1]$, i.e. $\|S[q_k,b] - P[1,b-q_k+1]\|_p^p$, is at most 2^k.*

Further, divide $S[q_k, b]$ into $\Theta(1/\varepsilon^p)$ blocks such that each block is either a single character, or the p'th moment of the difference between each block and the corresponding subword of $P[1, b-p_k+1]$ is at most $\varepsilon^p \cdot 2^k$. Let $q_k = q_k^0 \leq q_k^1 \leq \ldots q_k^{\ell_k} = b$ be the block borders. We choose $q_k^1, q_k^2, \ldots, q_k^{\ell_k}$ from left to right, and each position q_k^i is chosen to be the rightmost possible.

The L_p prefix encoding *of S is defined to contain sorted lists of the positions q_k and q_k^i, characters $S[q_k^i]$, and sketches for $(1 \pm C_p \cdot \varepsilon/p)$-approximating the p'th norm of $S[q_k^j, b]$, for all k, j and C_p as in Observation 5, see also Theorem 26.*

The encoding takes $\widetilde{\mathcal{O}}(\varepsilon^{-2-p}\log\sigma\log(\sigma n/\varepsilon)\log(n/\varepsilon))$ bits of space. We now show that given the L_p prefix encoding of a block B of the text of length b, one can compute a $(1\pm\varepsilon)$-approximation of the L_p distance between any prefix $P[1, b-j+1]$ of the pattern P and the corresponding suffix $B[j, b]$ of B.

▶ **Lemma 28.** *Let $1 < p \leq 2$. For any two vectors X, Y of equal length, $\left| \|X+Y\|_p^p - \|X\|_p^p \right| = \mathcal{O}(\|Y\|_p^p + \|Y\|_p \cdot \|X\|_p^{p-1})$.*

▶ **Lemma 29.** *Let $1 < p \leq 2$. Given the L_p prefix encoding of a block B of the text T of length b, one can find $(1\pm\varepsilon)$-approximation of the p'th moment of the difference between any prefix $P[1, b-j+1]$ of the pattern P and the corresponding suffix $B[j, b]$ of B in time $\widetilde{\mathcal{O}}(\varepsilon^{-2} + \log\sigma)$.*

Proof. Let q_k be the position that is closest to i from the left, and $q_k^i \le j < q_k^{i+1}$ (see Fig. 2). We can find q_k, q_k^i, q_k^{i+1} in time $\mathcal{O}(\log(b\sigma^p) + 1/\varepsilon^p)$ by iterating over the sorted lists.

The position q_k^{i+1} divides $P[1, b-j+1]$ into two parts, P_1 and P_2. Denote B_1 and B_2 the respective subwords of B they are aligned with (see Fig. 2). Let $m_1 = F_p(P_1 - B_1)$ and $m_2 = F_p(P_2 - B_2)$. Then $m = F_p(P[1, b-j+1] - B[j, b])$, being the value we need to approximate, is equal to $m_1 + m_2$.

We can find $m_2' \overset{\varepsilon}{=} m_2$ using the sketches for $B_2 = B[q_k^{i+1}, b]$ and P_2 in time $\widetilde{\mathcal{O}}(1/\varepsilon^2)$. Furthermore, if $q_k^i = q_k^{i+1} - 1$, then we can compute m_1 exactly as we store $B[q_k^i]$. Otherwise, we consider the subword $\tilde{P} = P[j - q_k + 1, q_k^{i+1} - q_k + 1]$ of the pattern P. Denote $m_1' = F_p(P_1 - \tilde{P})$ and use it as our estimation of m_1.

Since $1 \le p \le 2$, by definition, $F_p(B_1 - \tilde{P}) \le \varepsilon^p \cdot 2^{k-1}$, and $F_p(P_1 - B_1) \le 2^k$. By Lemma 28 with $X = P_1 - B_1$ and $Y = B_1 - \tilde{P}$,

$$|m_1' - m_1| = \mathcal{O}(\|B_1 - \tilde{P}\|_p^p + \|B_1 - \tilde{P}\|_p \|P_1 - B_1\|_p^{p-1}) = \mathcal{O}(\varepsilon^p 2^k + \varepsilon (2^k)^{\frac{1}{p}} (2^k)^{\frac{p-1}{p}}) = \mathcal{O}(\varepsilon 2^k)$$

and finally $|(m_1 + m_2) - (m_1' + m_2')| \le \mathcal{O}(\varepsilon m) + \varepsilon m_2 = \mathcal{O}(\varepsilon m)$. ◀

▶ **Lemma 30.** *Let $1 < p \le 2$. The L_p prefix encoding of a b-length block B of the text can be computed in time $\tilde{\mathcal{O}}(b^2 + \varepsilon^{-2} b \log \sigma)$ and space $\tilde{\mathcal{O}}(b + \varepsilon^{-2-p} \log^2 \sigma)$.*

Proof. For $j = 1, \ldots, b$, we naively compute the L_p distance between the suffix of B and the prefix of P in $\mathcal{O}(b)$ time. We then find the positions q_k. For each $k = 0, \ldots, \lceil \log b\sigma^p \rceil$, we can find the positions q_k^i in $\mathcal{O}(b)$ time and compute the sketches in $\tilde{\mathcal{O}}(\varepsilon^{-2} b)$ time by Theorem 26. ◀

3 Suffix sketches

In this section, we give the definitions and explain how we maintain the suffix sketches for each of the distances.

3.1 Hamming distance

We first recall Euclidean suffix sketches as presented in [17]. In fact, we will not use them for the Euclidean distance as for it we can use the generic solution of Section 3.3, but they will serve as a foundation of Hamming suffix sketches.

All sketches presented in this section are correct with constant probability, which can be amplified to $1 - \delta$ for arbitrarily small δ by a standard method of repeating sketching independently $\Theta(\log \delta^{-1})$ times and taking the median of the estimates.

▶ **Lemma 31** (Euclidean sketches [2])**.** *Let M be a random matrix of size $d \times n$ filled with 4-wise independent random ± 1 variables, for $d = \Theta(\varepsilon^{-2})$ chosen big enough. For a vector $X \in \mathbb{R}^n$ there is $\frac{1}{\sqrt{d}} \|MX\|_2 \overset{\varepsilon}{=} \|X\|_2$ with constant probability $9/10$, taken over all possible choices of M. We say that a vector MX of dimension d is a Euclidean sketch of X.*

▶ **Definition 32** (Euclidean suffix sketches [17])**.** *Consider a word X of length n. We define its Euclidean suffix sketch as follows.*

Let b be the block length*. Let $\mathcal{R}$ be a random matrix of size $d \times b$ filled with 4-wise independent random ± 1 variables and let $\alpha_1, \ldots, \alpha_{\lceil n/b \rceil}$ be 4-wise independent random coefficients with values ± 1 as well. We define a matrix M of size $d \times n$ such that $M_{i,jb+k} = \alpha_j \cdot \mathcal{R}_{i,k}$.*

Let X' be a word of length $\lceil n/b \rceil \cdot b$ obtained from X by appending an appropriate number of zeroes. The Euclidean suffix sketch *of X is defined as $\mathsf{eSketch}(X) = MX'$, where X' is considered as a vector.*

Observe that the matrix M does not need to be accessed explicitly. Indeed, from $MX' = \sum_i \alpha_i \cdot \mathcal{R} \cdot \left[X'[bi],\ \ldots,\ X'[bi+b-1]\right]^T$ it follows that the Euclidean suffix sketch can be computed by first sketching each block of X' using the matrix $\mathcal{R}$, and then taking a linear combination of the sketches of the blocks (using the random ± 1 coefficients α_i).

▶ **Lemma 33** ([17]). *Selecting $d = \Theta(\varepsilon^{-2})$ gives $\frac{1}{\sqrt{d}}\|\mathsf{eSketch}(X)\|_2 \overset{\varepsilon}{=} \|X\|_2$ with probability at least $9/10$ (taken over all possible choices of $\mathcal{R}, \alpha_i$).*

By linearity of sketches, we obtain $\|X - Y\|_2 \overset{\varepsilon}{=} \frac{1}{\sqrt{d}}\|\mathsf{eSketch}(X) - \mathsf{eSketch}(Y)\|_2$ with probability at least $9/10$ as well.

We now define Hamming suffix sketches. First note that for binary words X, Y there is $\mathsf{Ham}(X,Y) = \|X - Y\|_2$, and therefore in the case of the binary alphabet we can use the Euclidean suffix sketches. We will now show how to reduce the case of arbitrary polynomial-size alphabets to the case of the binary alphabet.

To this end, [17] used a random mapping of Karloff [29] as a black-box reduction, which led to sketches of size $\sim \varepsilon^{-4}$. We now show a more careful reduction to avoid this overhead and to achieve dependency ε^{-2} in total. Consider a word morphism defined on alphabet as $\mu : \Sigma \to \{0,1\}^\sigma$, $\mu(a) = 0^a 10^{\sigma-a-1}$ (and acting on words by concatenating the images of each character of the input word). Note that $\|\mu(X) - \mu(Y)\|_2^2 = 2 \cdot \|X - Y\|_H$, thus using the Euclidean suffix sketches on top of $\mu(X)$ and $\mu(Y)$ allows computation of the respective Hamming distance. Formally,

▶ **Definition 34** (Hamming suffix sketches [17]). *Consider a word X of length n on the alphabet of size σ. We define its Hamming suffix sketch as follows.*

Let b be the block length*, $\mathcal{R}$ be a random matrix of size $d \times \sigma b$ filled with 4-wise independent random ± 1 variables, and $\alpha_1, \ldots, \alpha_{\lceil n/b \rceil}$ be 4-wise independent random coefficients with values ± 1 as well. We define a matrix M of size $d \times \sigma n$ such that $M_{i,\sigma j b+k} = \alpha_j \cdot \mathcal{R}_{i,k}$.*

Let X' be a word of length $\lceil n/b \rceil \cdot b$ obtained from X by appending an appropriate number of zeroes. The Hamming suffix sketch *of X is defined as $\mathsf{hSketch}(X) = M\mu(X')$, where $\mu(X')$ is considered as a vector.*

▶ **Lemma 35.** *Selecting $d = \Theta(\varepsilon^{-2})$ gives $\frac{1}{2d}\|\mathsf{hSketch}(X)\|_2^2 \overset{\varepsilon}{=} \|X\|_H$ with probability at least $9/10$ (taken over all possible choices of $\mathcal{R}, \alpha_i$).*

Proof. Follows immediately as a corollary of Lemma 33 and the properties of the embedding μ. In more detail, the following holds with probability at least $9/10$:

$$\frac{1}{2d} \cdot \|\mathsf{hSketch}(X)\|_2^2 =$$
$$= \frac{1}{2d}\|M\mu(X')\|_2^2 = \frac{1}{2d}\|M\mu(X)\|_2^2 = \frac{1}{2d}\|\mathsf{eSketch}(\mu(X))\|_2^2 \overset{\varepsilon}{=} \frac{1}{2}\|\mu(X)\|_2^2 = \|X\|_H. \qquad \blacktriangleleft$$

As $\mu(X), \mu(Y)$ are sparse, there is an efficient streaming algorithm for maintaining the Hamming suffix sketches of a text:

▶ **Lemma 36.** *Given a text T, there is a streaming algorithm that for every position i outputs the Hamming suffix sketch of a word $T[b \cdot k + 1, i]$, where k is the largest integer such that $i - b \cdot k \le n$. The algorithm takes $\mathcal{O}(dn/b + \log d\sigma n)$ space and $\mathcal{O}(d(1 + n/b^2))$ time per character.*

3.2 Manhattan (L_1) distance

To show efficient suffix sketches for the Manhattan distance, we consider a word morphism $\nu : \Sigma \to \{0,1\}^\sigma$, $\nu(a) = 1^a 0^{\sigma-a}$. Note that $\|\nu(X) - \nu(Y)\|_2^2 = \|\nu(X) - \nu(Y)\|_H = \|X - Y\|_1$, thus using the Hamming suffix sketches on top of $\nu(X)$ and $\nu(Y)$ allows computation of the respective Manhattan distance.

However, if we apply the morphism straightforwardly, we will have to pay an extra σ factor per character to compute the Manhattan suffix sketches. To improve the running time, we will use range-summable hash functions. Range-summable hash functions were introduced by Feigenbaum et al. [20], and later their construction was improved by Calderbank et al. [9].

▶ **Definition 37** (cf. [9]). *A family $\mathcal{H}$ of hash functions $h(x;\xi) : [t] \times \{0,1\}^s \to \{-1,1\}$ (here x is the argument and ξ is the seed) is called k-independent, range-summable if it satisfies the following properties for any $h \in \mathcal{H}$:*

1. (k-independent) *for all distinct $0 \le x_1, \ldots, x_k < t$ and all $b_1, \ldots, b_k \in \{-1,+1\}$,*

$$\Pr_{\xi \in \{0,1\}^s}[h(x_1;\xi) = b_1 \wedge \cdots \wedge h(x_k;\xi) = b_k] = 2^{-k}$$

2. (range-summable) *there exists a function g such that given a pair of integers $0 \le \alpha, \beta \le \sigma$, and a seed ξ, the value $g(\alpha,\beta;\xi) = \sum_{\alpha \le x < \beta} h(x;\xi)$ can be computed in time polynomial in $\log t$.*[3]

▶ **Corollary 38** (cf. Theorem 3.1 [9]). *There is a 4-independent, range-summable family of hash functions $h(x;\xi) : [t] \times \{0,1\}^s \to \{-1,+1\}$ with a random seed ξ of length $s = \mathcal{O}(\log^2 t)$ such that any range-sum $g(\alpha,\beta;\xi)$ can be computed in $\mathcal{O}(\log^3 t)$ time.*

▶ **Observation 39.** *For a word $X = x_1 x_2 \ldots x_n$, let $Y = \nu(X) = y_1 y_2 \ldots y_{n\sigma}$. Let h, g be as in Corollary 38 with $t = n\sigma$. Then $\sum_{i=1}^{n} g(i\sigma, i\sigma + x_i; \xi) = \sum_{i=1}^{n\sigma} y_i h(i;\xi)$.*

Thus, we see that range-summable hash functions can be used to efficiently simulate ν.

▶ **Definition 40** (Manhattan suffix sketches). *Consider X be a word of length n. We define its Manhattan suffix sketch as follows.*

Let b be the block length. *Let h, g be as in Corollary 38 with $t = bd\sigma$. Let $\mathcal{R}$ be a random matrix of size $d \times \sigma b$ filled with 4-wise independent random ± 1 variables, such that $\mathcal{R}_{i,k} = h(ib\sigma + k;\xi)$ and let $\alpha_1, \ldots, \alpha_{\lceil n/b \rceil}$ be 4-wise independent random coefficients with values ± 1 as well. We define a matrix M of size $d \times \sigma n$ such that $M_{i,\sigma jb+k} = \alpha_j \cdot \mathcal{R}_{i,k} = \alpha_j \cdot h(dk+i;\xi)$.*

Let X' be a word of length $\lceil n/b \rceil \cdot b$ obtained from X by appending an appropriate number of zeroes. The Manhattan suffix sketch *of X is defined as $\mathsf{mSketch}(X) = M\nu(X')$, where $\nu(X')$ is considered as a vector.*

▶ **Lemma 41.** *Selecting $d = \Theta(1/\varepsilon^2)$ gives $\frac{1}{d}\|\mathsf{mSketch}(X)\|_2^2 \overset{\varepsilon}{=} \|X\|_1$ with probability at least $9/10$ (taken over all possible choices of α_i and ξ).*

Proof. Follows immediately as a corollary of Lemma 33 and the properties of the embedding ν. In more detail, the following holds with probability at least $9/10$:

$$\frac{1}{d} \cdot \|\mathsf{mSketch}(X)\|_2^2 =$$
$$= \frac{1}{d}\|M\nu(X')\|_2^2 = \frac{1}{d}\|M\nu(X)\|_2^2 = \frac{1}{d}\|\mathsf{eSketch}(\nu(X))\|_2^2 \overset{\varepsilon}{=} \|\nu(X)\|_2^2 = \|X\|_1. \qquad \blacktriangleleft$$

[3] In [9], the function h was defined to take values in $\{0,1\}$. We can change the range of values to $\{-1,+1\}$ by taking $h' = 1 - 2h$ while preserving the properties.

▶ **Lemma 42.** *Given a text T, there is a streaming algorithm that for every position i outputs the Manhattan suffix sketch of a word $T[b \cdot k + 1, i]$, where k is the smallest integer such that $i - b \cdot k \leq n$. The algorithm takes $\mathcal{O}(d \cdot (n/b) + \log^2 \sigma)$ space, and $\mathcal{O}(d(1 + n/b^2) \cdot \log^3(bd\sigma))$ time per character.*

3.3 Generic (L_p) distance for $0 < p \leq 2$

For generic L_p distances, we use the approach of [26] based on p-stable distributions.

▶ **Corollary 43.** *Given a text T, there is a streaming algorithm that for every position i outputs the L_p suffix sketch of a word $T[b \cdot k + 1, i]$, where k is the smallest integer such that $i - b \cdot k \leq n$. The algorithm takes $\mathcal{O}(\varepsilon^{-2}(n/b) \cdot \log(\sigma n/\varepsilon) \log(n/\varepsilon))$ bits of space and $\mathcal{O}(\varepsilon^{-2}(n/b) \log(n))$ time per character.*

Proof. We start a new instance of the sketching algorithm of Theorem 26 at every block border and continue running it for the next $\lceil n/b \rceil$ blocks. At each moment, there are $\mathcal{O}(n/b)$ active instances of the algorithm. The bounds follow. ◀

4 Proof of Theorem 3

Recall the structure of the algorithms. During the preprocessing, we compute the suffix sketches of suffixes $P[1, n], P[2, n], \ldots, P[b, n]$ of P. During the main stage, the text is processed by blocks of length b. To compute an approximation of the distance / the p'th moment at a particular alignment, we divide the pattern into two parts: a prefix of length at most b, and the remaining suffix. We compute an approximation of the distance / the p'th moment for both of the parts and sum them up to obtain the final answer. To compute an approximation of the distance / the p'th moment between the prefix and the corresponding block of the text, we compute, while reading each block of the text, its prefix encoding, and to compute an approximation of the distance / the p'th moment between the suffix and the text, we use the suffix sketches.

1) **Hamming (L_0) distance.** When we receive a new block of the text, we compute its Hamming prefix encoding using the algorithm of Lemma 10 in $\mathcal{O}(b)$ space. We de-amortize the computation over the subsequent block and spend $\widetilde{\mathcal{O}}(\varepsilon^{-2})$ time per character. We store the resulting encoding for the next $\mathcal{O}(n/b)$ blocks. In total, the encodings require $\widetilde{\mathcal{O}}(\varepsilon^{-2}n/b)$ space. The Hamming suffix sketches of $P[1, n], P[2, n], \ldots, P[b, n]$ occupy $\mathcal{O}(\varepsilon^{-2}b)$ space. The algorithm of Lemma 36 that computes the suffix sketches takes $\mathcal{O}(\varepsilon^{-2}n/b + \log(\varepsilon^{-2}\sigma n))$ space and $\mathcal{O}(\varepsilon^{-2}(1 + n/b^2))$ time per character. Consider a block starting with position p. To compute the Hamming distances between n-length subwords that end in this block and the pattern, we apply the following approach. First, while reading the block preceding the current one, we decode the Hamming prefix encoding of the block that starts at position $p - n$ using Lemma 11. We de-amortize the algorithm to spend $\widetilde{\mathcal{O}}(\varepsilon^{-2})$ time per character. Hence, at the position i, we know the $(1 \pm \varepsilon)$-approximation between the prefixes of the pattern and the corresponding subwords of the text. At each position, we can compute the Hamming distance between the corresponding suffix of the pattern and the text in $\widetilde{\mathcal{O}}(\varepsilon^{-2})$ time using the Hamming suffix sketch. By taking $b = \sqrt{n}$, we obtain the claim.
2) **Manhattan (L_1) distance.** We proceed analogously to the Hamming distance case. The Manhattan prefix encoding of each block is computed using Lemma 16, in $\widetilde{\mathcal{O}}(b)$ time per character. We store the resulting encoding for the next $\mathcal{O}(n/b)$ blocks, giving

in total $\widetilde{\mathcal{O}}(\varepsilon^{-2}n/b)$ space. The Manhattan suffix sketches of $P[1,n], P[2,n], \ldots, P[b,n]$ occupy $\mathcal{O}(\varepsilon^{-2}b)$ space. Algorithm of Lemma 42 takes $\widetilde{\mathcal{O}}(\varepsilon^{-2}(b+n/b)+\log^2\sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-2}(1+n/b^2))$ time per character. For decoding the prefix encoding we use Lemma 17, spending $\widetilde{\mathcal{O}}(b)$ time per character. Once again we take $b=\sqrt{n}$, and assume w.l.o.g. $\varepsilon^{-1}\leq\sqrt{n}$ (as otherwise we can use a naive algorithm with $\mathcal{O}(n)$ space and $\mathcal{O}(n)$ time per character).

3) **Generic (L_p) distance for $0<p<1$.** The L_p prefix encodings of the blocks are computed using Lemma 23, using $\widetilde{\mathcal{O}}(t\cdot b)$ time per character. We store the resulting encodings for the next $\mathcal{O}(n/b)$ blocks, giving in total $\widetilde{\mathcal{O}}(\varepsilon^{-2}n/b)$ space. The L_p suffix sketches of $P[1,n], P[2,n], \ldots, P[b,n]$ occupy $\widetilde{\mathcal{O}}(\varepsilon^{-2}b\log\sigma)$ space. Algorithm of Corollary 43 computes the L_p suffix sketches for the text in $\widetilde{\mathcal{O}}(\varepsilon^{-2}(n/b)\log\sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-2}n/b)$ time per character. For decoding the prefix encoding we use Lemma 24, spending $\widetilde{\mathcal{O}}(t\cdot b)$ time per character. We take $b=\sqrt{n}$, and substitute t accordingly to Theorem 18.
4) **Generic (L_p) distance for $1<p<2$.** Note that for $\varepsilon<1/n$ we can use a naive algorithm, that is to store S itself in $\mathcal{O}(n)$ space. The update takes constant time, and computing the L_p norm takes $\mathcal{O}(n)$ time which is better than the guarantees of the theorem for such values of ε. For $\varepsilon\geq 1/n$, the algorithm of Lemma 30 computes the L_p prefix encodings of the blocks in $\widetilde{\mathcal{O}}(b+\varepsilon^{-2-p}\log^2\sigma)$ space and $\widetilde{\mathcal{O}}(b+\varepsilon^{-2}\log\sigma)$ time per character. The encodings occupy $\widetilde{\mathcal{O}}(\varepsilon^{-2-p}(n/b)\log^2\sigma)$ space. The L_p suffix sketches of $P[1,n], P[2,n], \ldots, P[b,n]$ occupy $\widetilde{\mathcal{O}}(\varepsilon^{-2}b\log\sigma)$ space. Algorithm of Corollary 43 computes the L_p suffix sketches for the text in $\widetilde{\mathcal{O}}(\varepsilon^{-2}(n/b)\log\sigma)$ space and $\widetilde{\mathcal{O}}(\varepsilon^{-2}n/b)$ time per character. Taking $b=\varepsilon^{-p/2}\sqrt{n}$ and assuming w.l.o.g. $\varepsilon^{-1}<\sqrt{n}$, we obtain the claim.

5 Conclusion

We pose several open questions. First is whether the time-complexity for $1/2<p<1$ can be improved to not involve any dependency on σ. For this we need a better technique than bounding variance of the embedding into Hamming distance: in our technique, the tail gets "too heavy". Another pressing question is whether for all values of $p>0$ we could improve upon $\sqrt{n}$ time per character. We also remark that it seems unlikely that an embedding to Hamming space could be used to reduce space complexity for $p>1$: L_p^p does not admit the triangle inequality while the Hamming distance does, and the L_p distance is not additive with respect to concatenation, while the Hamming distance is.

References

1 Karl R. Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987.

2 Dimitris Achlioptas. Database-friendly random projections: Johnson–Lindenstrauss with binary coins. *J. Comput. Syst. Sci.*, 66(4):671–687, 2003. doi:10.1016/S0022-0000(03)00025-4.

3 Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM 2002*, pages 1–10, 2002. doi:10.1007/3-540-45726-7_1.

4 Vladimir Braverman, Ran Gelles, and Rafail Ostrovsky. How to catch L_2-heavy-hitters on sliding windows. *Theor. Comput. Sci.*, 554:82–94, 2014.

5 Vladimir Braverman and Rafail Ostrovsky. Smooth histograms for sliding windows. In *FOCS 2007*, pages 283–293, 2007.

6 Vladimir Braverman and Rafail Ostrovsky. Effective computations on sliding windows. *SIAM J. Comput.*, 39(6):2113–2131, 2010.

7 Vladimir Braverman, Rafail Ostrovsky, and Alan Roytman. Zero-one laws for sliding windows and universal sketches. In *APPROX-RANDOM 2015*, pages 573–590, 2015.

8 Vladimir Braverman, Rafail Ostrovsky, and Carlo Zaniolo. Optimal sampling from sliding windows. *J. Comput. Syst. Sci.*, 78(1):260–272, 2012.

9 A. Robert Calderbank, Anna C. Gilbert, Kirill Levchenko, S. Muthukrishnan, and Martin Strauss. Improved range-summable random variable construction algorithms. In *SODA 2005*, pages 840–849, 2005.

10 Amit Chakrabarti and Oded Regev. An optimal lower bound on the communication complexity of gap-Hamming-distance. In *STOC 2011*, pages 51–60, 2011. doi:10.1145/1993636.1993644.

11 J. M. Chambers, C. L. Mallows, and B. W. Stuck. A method for simulating stable random variables. *J. of the American Statistical Association*, 71(354):340–344, 1976.

12 Timothy M. Chan, Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, and Ely Porat. Approximating text-to-pattern Hamming distances. In *STOC 2020*, pages 643–656, 2020. doi:10.1145/3357713.3384266.

13 Peter Clifford, Raphaël Clifford, and Costas S. Iliopoulos. Faster algorithms for delta, gamma-matching and related problems. In *CPM 2005*, pages 68–78, 2005. doi:10.1007/11496656_7.

14 Raphaël Clifford, Klim Efremenko, Ely Porat, and Amir Rothschild. From coding theory to efficient pattern matching. In *SODA 2009*, pages 778–784, 2009.

15 Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. The k-mismatch problem revisited. In *SODA 2016*, pages 2039–2052, 2016. doi:10.1137/1.9781611974331.ch142.

16 Raphaël Clifford, Tomasz Kociumaka, and Ely Porat. The streaming k-mismatch problem. In *SODA 2019*, pages 1106–1125, 2019. doi:10.1137/1.9781611975482.68.

17 Raphaël Clifford and Tatiana Starikovskaya. Approximate Hamming distance in a stream. In *ICALP 2016*, pages 20:1–20:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.20.

18 Mayur Datar, Aristides Gionis, Piotr Indyk, and Rajeev Motwani. Maintaining stream statistics over sliding windows. *SIAM J. Comput.*, 31(6):1794–1813, 2002. doi:10.1137/S0097539701398363.

19 Marianne Durand and Philippe Flajolet. Loglog counting of large cardinalities (extended abstract). In *ESA 2003*, pages 605–617, 2003. doi:10.1007/978-3-540-39658-1_55.

20 Joan Feigenbaum, Sampath Kannan, Martin Strauss, and Mahesh Viswanathan. An approximate L1-difference algorithm for massive data streams. *SIAM J. Comput.*, 32(1):131–151, 2002. doi:10.1137/S0097539799361701.

21 Michael J. Fischer and Michael S. Paterson. String-matching and other products. Technical report, Massachusetts Institute of Technology, 1974.

22 Sumit Ganguly. Counting distinct items over update streams. *Theor. Comput. Sci.*, 378(3):211–222, 2007. doi:10.1016/j.tcs.2007.02.031.

23 Paweł Gawrychowski and Przemysław Uznański. Towards unified approximate pattern matching for Hamming and L_1 distance. In *ICALP 2018*, pages 62:1–62:13, 2018. doi:10.4230/LIPIcs.ICALP.2018.62.

24 Phillip B. Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *SPAA 2001*, pages 281–291, 2001. doi:10.1145/378580.378687.

25 Shay Golan, Tsvi Kopelowitz, and Ely Porat. Towards optimal approximate streaming pattern matching by matching multiple patterns in multiple streams. In *ICALP 2018*, pages 65:1–65:16, 2018.

26 Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, 2006. doi:10.1145/1147954.1147955.

27 Thathachar S. Jayram and David P. Woodruff. Optimal bounds for Johnson-Lindenstrauss transforms and streaming problems with subconstant error. *ACM Trans. Algorithms*, 9(3):26, 2013.

28 Daniel M. Kane, Jelani Nelson, and David P. Woodruff. An optimal algorithm for the distinct elements problem. In *PODS 2010*, pages 41–52, 2010. doi:10.1145/1807085.1807094.

29 Howard J. Karloff. Fast algorithms for approximately counting mismatches. *Inf. Process. Lett.*, 48(2):53–60, 1993. doi:10.1016/0020-0190(93)90177-B.

30 Lev Klebanov and Andrei Yakovlev. How high is the level of technical noise in microarray data? *Biol Direct.*, 2:9, April 2007. doi:10.1186/1745-6150-2-9.

31 Tsvi Kopelowitz and Ely Porat. Breaking the variance: Approximating the Hamming distance in $1/\epsilon$ time per alignment. In *FOCS 2015*, pages 601–613, 2015. doi:10.1109/FOCS.2015.43.

32 Tsvi Kopelowitz and Ely Porat. A simple algorithm for approximating the text-to-pattern Hamming distance. In *SOSA@SODA 2018*, pages 10:1–10:5, 2018. doi:10.4230/OASIcs.SOSA.2018.10.

33 S. R. Kosaraju. Efficient string matching. Manuscript, 1987.

34 Karim Labib, Przemysław Uznański, and Daniel Wolleb-Graf. Hamming distance completeness. In *CPM 2019*, pages 14:1–14:17, 2018. doi:10.4230/LIPIcs.CPM.2019.14.

35 Ohad Lipsky and Ely Porat. L_1 pattern matching lower bound. *Inf. Process. Lett.*, 105(4):141–143, 2008. doi:10.1016/j.ipl.2007.08.011.

36 Noam Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12:4:449–461, 1992. doi:10.1007/BF01305237.

37 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.

38 Benny Porat and Ely Porat. Exact and approximate pattern matching in the streaming model. In *FOCS 2009*, pages 315–323, 2009. doi:10.1109/FOCS.2009.11.

39 Jan Studený and Przemysław Uznański. Approximating approximate pattern matching. In *CPM 2019*, volume 128, pages 15:1–15:13, 2019. doi:10.4230/LIPIcs.CPM.2019.15.

40 Vladimir M. Zolotarev. *One-dimensional stable distributions*, volume 65 of *Translations of Mathematical Monographs*. American Mathematical Soc., 1986. Translated from Russian by H.H. McFaden, Translation edited by B. Silver.

6 Omitted proofs

▶ **Lemma 4.** *Let $2 \leq 1/\varepsilon < n$ and $0 \leq p \leq 2$. Any $(1 \pm \varepsilon)$-approximation algorithm that computes the L_p distance between a pattern and a streaming text for each alignment, must use $\Omega(\min(1/\varepsilon^2, n))$ bits of space.*

Proof. Let us first show the lower bound for $p = 0$, i.e., for Hamming distance. We show the lower bound by reduction to a two-party communication complexity problem called GAP-Hamming-distance. In this problem, the two parties, Alice and Bob are given two binary words of length n and a parameter $g = \varepsilon n$, $1 \leq g \leq n/2$. Alice sends Bob a message, and Bob's task is to output 1 if the Hamming distance between his and Alice's word is larger than $n/2 + g$, and zero if it is at most $n/2 - g$. Otherwise, he can output "don't know". By Proposition 4.4 [10], the communication complexity of this problem is $\Omega(\min\{1/\varepsilon^2, n\})$.

We can now show a space lower bound for any $(1\pm\varepsilon)$-approximate algorithm for computing the Hamming distance between the pattern and the text by a standard reduction. Suppose that $2 \leq 1/\varepsilon \leq n$ there is an algorithm that uses $o(\min\{1/\varepsilon^2, n\})$ bits of space. Let P be Alice's word, T Bob's word. After reading P, the algorithm stores all the information about it in $o(\min\{1/\varepsilon^2, n\})$ bits of space. We construct the communication protocol as follows: Alice sends the information about P to Bob. Using it, Bob can continue running the algorithm and compute the approximation of the Hamming distance between P and T. We have thus developed a communication protocol with complexity $o(\min\{1/\varepsilon^2, n\})$, a contradiction.

We can now show the lower bound for $0 < p \leq 2$. We immediately obtain a space lower bound for any $(1 \pm \varepsilon)$-approximate algorithm for computing the p'th moment between the pattern and the text at every alignment. Indeed, on binary words the p'th moment is equal to the Hamming distance for all $0 < p \leq 2$. The lower bound for the L_p distance follows by Observation 5. ◀

▶ **Lemma 8.** *For Z_f as in Algorihtm 7 there is $Z_f \overset{\varepsilon}{=} \|U-V\|_H$ with probability at least $3/4$.*

Proof. Denote $m = \|U-V\|_H$. Consider a fixed value r. Let $I_1, I_2, \ldots, I_n$ be binary variables indicating existence of a mismatch between $\mathsf{hSub}_r(U)$ and V at positions $1, \ldots, n$, so that $X_r = \sum_j I_j$. We observe that $\mathbb{E}[X_r] = m/2^r$ and therefore $\mathbb{E}[Z_r] = m$, because each of the m positions with mismatch between U and V generates a mismatch between $\mathsf{hSub}_r(U)$ and V with probability $1/2^r$.

Furthermore, as the function h in Definition 6 is drawn from a pairwise independent family, there is $\mathrm{Var}[X_r] = \sum_j \mathrm{Var}[I_j] \le \sum_j \mathbb{E}\left[(I_j)^2\right] = \sum_j \mathbb{E}[I_j] = \mathbb{E}[X_r] = m/2^r$. Let $c = \min\{i : \mathbb{E}[X_i] \le k\} = \lceil \log_2\left(\frac{m}{k}\right)\rceil$. By Chebyshev's inequality, we have

$$\Pr[|Z_r - m| \ge 4\sqrt{m2(c+1)}] = \Pr[|X_r - m/2^r| \ge 2^{2+(c+1-r)/2}\sqrt{m/2^r}] \le 1/2^{4+(c+1-r)} \quad (2)$$

We estimate $\Pr[f > c+1] = \Pr[X_{c+1} > k]$. Assume w.l.o.g. that $k \ge 32$. Observe that $m/2^c \le k$, which implies, for $k \ge 32$, $m/2^{c+1} + 4\sqrt{m/2^{c+1}} \le k/2 + 4\sqrt{k/2} \le k$. By Equation 2, there is

$$\Pr[X_{c+1} > k] \le \Pr[X_{c+1} \ge m/2^{c+1} + 4\sqrt{m/2^{c+1}}] \le 1/16.$$

It follows that $\Pr[f > c+1] = \Pr[X_{c+1} > k] \le 1/16$. Hence, we obtain

$$\begin{aligned}
\Pr[|Z_f - m| \ge 4\sqrt{2/k}\cdot m] &\le \Pr[|Z_f - m| \ge 4\sqrt{m2^{c+1}}] \le \\
&\le \Pr[f > c+1] + \sum_{r=0}^{c+1} \Pr[|Z_f - m| \ge 4\sqrt{m2^{c+1}} \text{ and } f = r] \le \\
&\le \Pr[f > c+1] + \sum_{r=0}^{c+1} \Pr[|Z_r - m| \ge 4\sqrt{m2^{c+1}}] \le \\
&\le 1/16 + \sum_{r=1}^{c+1} 1/2^{4+(c+1-r)} < 1/4.
\end{aligned}$$

It follows that we can choose $k = \Theta(1/\varepsilon^2)$ large enough so that $Z_f \overset{\varepsilon}{=} \|U-V\|_H$ with probability $\ge 3/4$. ◀

▶ **Lemma 14.** *For Z_f as in Algorihtm 13 there is $Z_f \overset{\varepsilon}{=} \|U-V\|_1$ with probability $\ge 3/4$.*

Proof. Take some position i and denote for short $a = \mathsf{mSub}_r(U)[i]$ and $b = \mathsf{mSub}_r(V)[i]$ and $c = \frac{U[i]-V[i]}{2^r}$. There is $|a-b| \in \left\{\lfloor|c|\rfloor, \lceil|c|\rceil\right\}$ and $\mathbb{E}[|a-b|] = |c|$. Since $|a-b| - \lfloor|c|\rfloor$ is a 0/1 variable, there is $\mathrm{Var}[|a-b|] = \mathrm{Var}\left[(|a-b| - \lfloor|c|\rfloor)\right] \le \mathbb{E}\left[(|a-b| - \lfloor|c|\rfloor)\right] \le \mathbb{E}[|a-b|]$. Summing for all values of i, we reach that

$$\mathrm{Var}[X_r] = \mathrm{Var}[\|\mathsf{mSub}_r(U) - \mathsf{mSub}_r(V)\|_1] \le \mathbb{E}[\|\mathsf{mSub}_r(U) - \mathsf{mSub}_r(V)\|_1] = \mathbb{E}[X_r].$$

Since we have reached an identical variance bound, the proof follows step-by-step the proof of Lemma 8. ◀

▶ **Theorem 18.** *Given $0 < p < 1$ and $\varepsilon > 0$ there is a word morphism $\varphi : c \in \Sigma \to c_0^{d_0} c_2^{d_2} \ldots c_{t-1}^{d_{t-1}}$ such that:*

1) *$t = \widetilde{\mathcal{O}}(\varepsilon^{-2})$ when $0 < p < 1/2$, $t = \widetilde{\mathcal{O}}(\varepsilon^{-3})$ when $p = 1/2$ and $t = \widetilde{\mathcal{O}}(\sigma^{\frac{2p-1}{1-p}}/\varepsilon^{2+3\cdot\frac{2p-1}{1-p}})$ when $1/2 < p < 1$.*
2) *values of t and $d_0, \ldots, d_{t-1}$ do not depend on c,*

3) *there exists a constant $\alpha = \alpha(p, \varepsilon)$ such that for any two words U, V of length at most n, we have $\|U - V\|_p^p \overset{\varepsilon}{=} \alpha \cdot \|\varphi(U) - \varphi(V)\|_H$ with probability at least $9/10$,*
4) *it is enough for the randomness to be realized by a hash function $r : [t] \to [D]$ from a 4-independent hash function family for some $D = poly(n\sigma\varepsilon^{-1})$, which can be generated from a $\widetilde{\mathcal{O}}(\log \sigma)$ bits size seed.*

Proof. We will consider three cases: $0 < p < 1/2$, $p = 1/2$, and $1/2 < p < 1$.

Case $0 < p < 1/2$. Our plan is to build upon the scheme highlighted earlier in this section. Specifically, we preserve the values of c_i.

Consider a pair of characters c, c'. First, note that $\mathbb{E}[x]$ is an increasing function of $|c - c'|$. From this and Equation 1 we obtain that $\mathbb{E}[x] \overset{\varepsilon}{=} |c - c'|^p \left(\frac{(1+\varepsilon)^p}{(1+\varepsilon)^p - 1} + \frac{1}{(1+\varepsilon)^{1-p} - 1} \right) \varepsilon^{-1}$ for all values of $|c - c'|$.

Second, fix $q = \lceil \frac{1}{1-p} \log_{1+\varepsilon}(\sigma\varepsilon^{-3}) \rceil$ and observe that truncating the sum after the $(q-1)$-th term introduces an additional factor $1 \pm \Theta(\varepsilon)$ to the approximation, since for $c \neq c'$ we have

$$\sum_{i \geq q} d_i \cdot \frac{|c - c'|}{(1+\varepsilon)^i} = \varepsilon^{-1}|c - c'| \frac{((1+\varepsilon)^q)^{p-1}}{1 - (1+\varepsilon)^{p-1}} \leq \frac{\varepsilon^{-1}\sigma}{(1 - (1+\varepsilon)^{p-1})\sigma\varepsilon^{-3}} = \Theta(\varepsilon).$$

We also round d_i down to the nearest integer, which introduces an additional $1 \pm \Theta(\varepsilon)$ relative error, since $\forall_i d_i \geq \varepsilon^{-1}$. Finally, we set $\varphi(c) = c_0^{d_0} \dots c_{q-1}^{d_{q-1}}$. We then have $\mathbb{E}[\|\varphi(c) - \varphi(c')\|_H] = \Theta(\varepsilon^{-2}|c - c'|^p \frac{1}{p(1-p)})$.

To guarantee that the equality holds with probability at least $9/10$ and not just in expectation, we repeat the scheme several times, with independent random seeds. That is, consider morphisms $\varphi_1(c), \varphi_2(c), \dots, \varphi_s(c)$ and define a morphism $\varphi(c) = \varphi_1(c)\varphi_2(c) \dots \varphi_s(c)$ with property:

$$\mathbb{E}[\|\varphi(c) - \varphi(c')\|_H] = s \cdot \mathbb{E}[\|\varphi_i(c) - \varphi_i(c')\|_H] = s \cdot \Theta(\varepsilon^{-2}|c - c'|^p \frac{1}{p(1-p)}).$$

Assume w.l.o.g. that $(1+\varepsilon)^{\ell-1} < |c - c'| \leq (1+\varepsilon)^\ell$. We proceed to bound

$$\begin{aligned}
\mathrm{Var}[\|\varphi(c) - \varphi(c')\|_H] &\leq s \cdot \sum_{i=\ell+1}^{q} (d_i)^2 \cdot \Pr[c_i \neq c_i'] \leq \\
&\leq s \cdot \sum_{i=\ell+1}^{q} \varepsilon^{-2}((1+\varepsilon)^{2p})^i \frac{|c - c'|}{(1+\varepsilon)^i} \leq \\
&\leq s \cdot \varepsilon^{-2}|c - c'| \sum_{i=\ell+1}^{\infty} ((1+\varepsilon)^{2p-1})^i \leq \\
&\leq s \cdot \varepsilon^{-2}|c - c'|^{2p} \frac{(1+\varepsilon)^{2p-1}}{1 - (1+\varepsilon)^{2p-1}} \leq \\
&= s \cdot \mathcal{O}(|c - c'|^{2p}\varepsilon^{-3} \frac{1}{1-2p}).
\end{aligned}$$

We set $s = \Theta(\frac{|c-c'|^{2p}\varepsilon^{-3}(p(p-1))^2}{\varepsilon^2(|c-c'|^p\varepsilon^{-2})^2(1-2p)}) = \mathcal{O}(\varepsilon^{-1}\frac{1}{1-2p})$ for the claim to hold via Chebyshev's inequality. The error probability coming from Chebyshev's inequality can be made arbitrarily small constant by fixing the constant factor in s to be large enough. We finally set $t = sq$.

Case $p = 1/2$. Note that for p, p' such that $|p - p'| \leq \log_\sigma(1+\varepsilon)$ we have $|x|^p \overset{\varepsilon}{=} |x|^{p'}$ for all $-\sigma \leq x \leq \sigma$. We can therefore reduce this case to $p = 1/2 - \log_\sigma(1+\varepsilon)$. However, we have to take into account that the asymptotic growth of t hides $1/(1-2p)$ dependency on p for $0 < p < 1/2$, hence $t = \widetilde{\mathcal{O}}(\varepsilon^{-3})$ for $p = 1/2$.

Case $1/2 < p < 1$. The proof follows the steps of the case $0 < p < 1/2$. We first bound the variance:

$$\begin{aligned}\operatorname{Var}\left[\|\varphi(c) - \varphi(c')\|_H\right] &\leq s \cdot \sum_{i=\ell+1}^{q} (d_i)^2 \cdot \Pr[c_i \neq c'_i] = \\ &= s \cdot \varepsilon^{-2}|c - c'| \sum_{i=\ell+1}^{q} ((1+\varepsilon)^{2p-1})^i = \\ &= s \cdot \mathcal{O}(\varepsilon^{-3}|c - c'|((1+\varepsilon)^q)^{2p-1}) = \\ &= s \cdot \mathcal{O}(\varepsilon^{-3}|c - c'|\sigma^{\frac{2p-1}{1-p}}/\varepsilon^{3\cdot\frac{2p-1}{1-p}}).\end{aligned}$$

We set $s = \Theta\left(\frac{\varepsilon^{-3}|c-c'|\sigma^{\frac{2p-1}{1-p}}/\varepsilon^{3\cdot\frac{2p-1}{1-p}}}{\varepsilon^{-2}|c-c'|^{2p}}\right) = \mathcal{O}(\sigma^{\frac{2p-1}{1-p}}/\varepsilon^{1+3\cdot\frac{2p-1}{1-p}})$, so that by Chebyshev's inequality, the probability of obtaining $\|U - V\|_p^p \overset{\varepsilon}{=} \alpha \cdot \|\varphi(U) - \varphi(V)\|_H$ is an arbitrarily small constant (by setting s to be large enough).

Randomness. The only source of randomness in the description are the values $r_i \in [0,1]$ picked u.a.r. and independently. We note that the values r_i can be picked instead as a finite precision floating-point numbers. Since all the values we are working with are bounded by $\operatorname{poly}(n\sigma\varepsilon^{-1})$, it is enough to set precision accordingly. We also observe that our concentration argument involves only Chebyshev's inequality and thus only the variance and the expected value, so it suffices to require that r_i are 4-wise independent. ◀

▶ **Lemma 21.** *For Z_f as in Algorihtm 20 there is $Z_f \overset{\varepsilon}{=} \|S - Q\|_H$ with probability $\geq 3/4$.*

Proof. Consider a fixed subsampling level r. For simplicity, let $\mathsf{Sub}_r(S) = s_1^{e'_1} s_2^{e'_2} \dots s_m^{e'_m}$ and $\mathsf{Sub}_r(Q) = q_1^{e'_1} q_2^{e'_2} \dots q_m^{e'_m}$. Define a random variable x_i to be the contribution of of $s_i^{e'_i}, q_i^{e'_i}$ to the Hamming distance X_r, i.e.

$$x_i = \|s_i^{e'_i} - q_i^{e'_i}\|_H = e'_i \cdot \|s_i - q_i\|_H.$$

Since $e'_i \in \{\lceil e_i/2^r \rceil, \lfloor e_i/2^r \rfloor\}$ and $\mathbb{E}[e'_i] = e_i/2^r$, we have $\mathbb{E}[x_i] = e_i \cdot \|s_i - q_i\|_H$ and

$$\operatorname{Var}[x_i] = \operatorname{Var}[x_i - \lfloor e_i/2^r \rfloor] \leq \mathbb{E}[x_i - \lfloor e_i/2^r \rfloor] \leq \mathbb{E}[x_i].$$

Summing over all values of i, we reach $\mathbb{E}[X_r] = \|S - Q\|_H$ and $\operatorname{Var}[X_r] \leq \mathbb{E}[X_r]$. These bounds are identical to that of Lemma 8 and we can proceed in a similar fashion to obtain the claim. ◀

▶ **Lemma 28.** *Let $1 < p \leq 2$. For any two vectors X, Y of equal length, $\left|\|X+Y\|_p^p - \|X\|_p^p\right| = \mathcal{O}(\|Y\|_p^p + \|Y\|_p \cdot \|X\|_p^{p-1})$.*

Proof. Consider $x, y \in \mathbb{R}$. If $|x| \geq |y|$, then by Taylor expansion, $|x+y|^p = |x|^p(1+y/|x|)^p = |x|^p(1 + \mathcal{O}(|y/x|)) = |x|^p \pm \mathcal{O}(|y||x|^{p-1})$. If $|x| < |y|$, then $|x+y|^p = \mathcal{O}(|y|^p)$. Thus for any real values, we have

$$|x+y|^p = |x|^p + \mathcal{O}(|y|^p + |y| \cdot |x|^{p-1}).$$

Denote $X = [x_1, x_2, \ldots, x_n]^T$ and $Y = [y_1, y_2, \ldots, y_n]^T$. There is

$$\|X+Y\|_p^p = \sum_i |x_i + y_i|^p = \sum_i |x_i|^p \pm \mathcal{O}\left(\sum_i |y_i|^p + \sum_i |y_i||x_i|^{p-1}\right).$$

Pick $q = p/(p-1)$ so that $1/p + 1/q = 1$. By Hölder's inequality:

$$\sum_i |y_i||x_i|^{p-1} \leq \left(\sum_i |y_i|^p\right)^{1/p} \left(\sum_i |x_i|^{(p-1)q}\right)^{1/q} = \|Y\|_p \|X\|_p^{p-1}. \quad \blacktriangleleft$$

▶ **Lemma 36.** *Given a text T, there is a streaming algorithm that for every position i outputs the Hamming suffix sketch of a word $T[b \cdot k + 1, i]$, where k is the largest integer such that $i - b \cdot k \leq n$. The algorithm takes $\mathcal{O}(dn/b + \log d\sigma n)$ space and $\mathcal{O}(d(1 + n/b^2))$ time per character.*

Proof. We fix the matrix $\mathcal{R}$ and the random coefficients $\alpha_1, \ldots, \alpha_{\lceil n/b \rceil}$ from Definition 34. We do not store $\mathcal{R}$ and α_i explicitly, but generate them using two hash functions drawn at random from a 4-wise independent family. For example, to generate $\mathcal{R}$ we can consider a family of polynomials $2((ax^3 + bx^2 + cx + d \bmod p) \bmod 2) - 1$, with parameters a, b, c, d chosen u.a.r. from the prime field $\mathbb{F}_p$ for $p \geq db$, and α_i can be generated in a similar fashion. This way, we need to store only $\mathcal{O}(\log(d\sigma b) + \log(n/b)) = \mathcal{O}(\log d\sigma n)$ random bits that define the coefficients of two polynomials to generate $\mathcal{R}$ and α_i.

We process the text T by blocks $B_1, B_2, \ldots$ of length b. For each block B_k we compute its sketch using the matrix $\mathcal{R}$. That is, at the beginning of each block we initialize its sketch as a zero vector of length d. When a new character $T[i]$ of a block B_k arrives, we compute and add $[M[1, i \cdot b\sigma + T[i]], M[2, i \cdot b\sigma + T[i]], \ldots, M[d, i \cdot b\sigma + T[i]]]^T$ to the sketch, which takes $\mathcal{O}(d)$ time. We store the sketch of B_k until the block $B_{k+\lceil n/b \rceil}$ and use it to compute the suffix sketches for the positions in this block.

Consider now a block $B_{k+\lceil n/b \rceil}$. We first compute the suffix sketch for the position $b \cdot (k + \lceil n/b \rceil)$, which is the position preceding the block $B_{k+\lceil n/b \rceil}$. The suffix sketch for it is simply a linear combination of the sketches of the blocks $B_{k+\lceil n/b \rceil - 1}, B_{k+\lceil n/b \rceil - 2}, \ldots, B_k$ with coefficients $\alpha_1, \ldots, \alpha_{\lceil n/b \rceil - 1}$. Since each sketch is a vector of length d, we can compute the linear combination in $\mathcal{O}(dn/b)$ time. To make this computation time-efficient, we start it b positions before position $b \cdot (k + \lceil n/b \rceil)$ arrives, and de-amortise the computation over these b positions. This way, we use only $\mathcal{O}(dn/b^2)$ time per character.

Now, using the suffix sketch for the position $b \cdot (k + \lceil n/b \rceil)$, we can compute the suffix sketches for all positions in the block $B_{k+\lceil n/b \rceil}$ one-by-one, using only $\mathcal{O}(d)$ time per character: When a new character $T[i]$ arrives, we add $[\alpha_{\lceil n/b \rceil} M[1, i \cdot b\sigma + T[i]], \alpha_{\lceil n/b \rceil} M[2, i \cdot b\sigma + T[i]], \ldots, \alpha_{\lceil n/b \rceil} M[d, i \cdot b\sigma + T[i]]]^T$ to the suffix sketch to update it.

Note that at any time we store $\mathcal{O}(n/b)$ sketches of the blocks, so the algorithm uses $\mathcal{O}(dn/b + \log d\sigma n)$ space in total. $\blacktriangleleft$

▶ **Lemma 42.** *Given a text T, there is a streaming algorithm that for every position i outputs the Manhattan suffix sketch of a word $T[b \cdot k + 1, i]$, where k is the smallest integer such that $i - b \cdot k \leq n$. The algorithm takes $\mathcal{O}(d \cdot (n/b) + \log^2 \sigma)$ space, and $\mathcal{O}(d(1 + n/b^2) \cdot \log^3(bd\sigma))$ time per character.*

Proof. The proof mirrors the proof of Lemma 36, and we describe the key elements. We fix the random coefficients $\alpha_1, \ldots, \alpha_{\lceil n/b \rceil}$ and the hash function h from Definition 40. As previously, we do not store the coefficients α_i explicitly, but generate them using a hash

function drawn at random from a 4-wise independent family. The matrix $\mathcal{R}$ is already defined by h, with the following parameters: it requires $\mathcal{O}(\log^2(bd\sigma))$ bits of seed, and range-sum queries are answered in time $\mathcal{O}(\log^3(bd\sigma))$.

In the sketching of blocks, we proceed in the same manner, except that when a new character $T[i]$ of a block B_k arrives, we compute and add $\sum_{0\leq j<T[i]}[M[1, i\cdot b\sigma+j], \ldots, M[d, i\cdot b\sigma + j]]^T = \alpha_i \cdot [g(b\sigma, b\sigma + T[i]; \xi), g(2b\sigma, 2b\sigma + T[i]; \xi), \ldots, g(d \cdot b\sigma, d \cdot b\sigma + T[i]; \xi)]^T$ to the sketch, which takes $\mathcal{O}(d \cdot \log^3(bd\sigma))$ time ($\log^3(bd\sigma)$ times slower as the corresponding step in Lemma 36).

Consider now a block $B_{k+\lceil n/b\rceil}$. When a new character $T[i]$ arrives, we update the suffix sketch by adding $\alpha_{\lceil n/b\rceil} \cdot [g(b\sigma, b\sigma + T[i]; \xi), g(2b\sigma, 2b\sigma + T[i]; \xi), \ldots, g(d \cdot b\sigma, d \cdot b\sigma + T[i]; \xi)]^T$ to it.

All of the operations are $\mathcal{O}(\log^3(bd\sigma))$ time slower than the corresponding steps in Lemma 36, and the memory complexity is increased by the seed size $\mathcal{O}(\log^2(bd\sigma))$ term ($\log^2 b$ and $\log^2 d$ terms get absorbed). ◀

Computing Bi-Lipschitz Outlier Embeddings into the Line

Karine Chubarian
Department of Mathematics, Statistics and Computer Science,
University of Illinois at Chicago, IL, USA
kchuba2@uic.edu

Anastasios Sidiropoulos
Department of Computer Science, University of Illinois at Chicago, IL, USA
sidiropo@uic.edu

Abstract

The problem of computing a bi-Lipschitz embedding of a graphical metric into the line with minimum distortion has received a lot of attention. The best-known approximation algorithm computes an embedding with distortion $O(c^2)$, where c denotes the optimal distortion [Bădoiu et al. 2005]. We present a bi-criteria approximation algorithm that extends the above results to the setting of *outliers*.

Specifically, we say that a metric space (X, ρ) admits a (k, c)-embedding if there exists $K \subset X$, with $|K| = k$, such that $(X \setminus K, \rho)$ admits an embedding into the line with distortion at most c. Given $k \geq 0$, and a metric space that admits a (k, c)-embedding, for some $c \geq 1$, our algorithm computes a $(\mathsf{poly}(k, c, \log n), \mathsf{poly}(c))$-embedding in polynomial time. This is the first algorithmic result for outlier bi-Lipschitz embeddings. Prior to our work, comparable outlier embeddings where known only for the case of additive distortion.

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

Keywords and phrases metric embeddings, outliers, distortion, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.36

Category APPROX

Funding *Anastasios Sidiropoulos*: Supported by NSF grants CCF-1815145, CCF-1934915, and by NSF CAREER award 1453472.

1 Introduction

The theory of metric embeddings provides an extensive toolbox that has found applications in several geometric data-analytic tasks. At the high level, an embedding of a metric space $\mathcal{M} = (X, \rho)$ into some metric space $\mathcal{M}' = (X', \rho)$ is a mapping $f : X \to X'$ that preserves certain interesting geometric properties of $\mathcal{M}$. In most cases, it is desirable to obtain embeddings that minimize some notion of *distortion*.

Despite the success of metric embeddings methods in several application domains, one significant limitation of most existing methods is that they are not robust to noise in the form of outlier points in the input. This setting is of particular interest in the case where the data does not perfectly fit the underlying geometric model, or when some points are corrupted due to measurement errors. The outlier model also has connections to the setting of adversarial machine learning [13]. More specifically, in the setting of *poisoning attacks*, it is often assumed that a small subset of the training data set is corrupted adversarially. For example, in a classification application, some of the training samples can be modified arbitrarily. Therefore, it is important to design data-analytic primitives that are robust against this type of adversarial input perturbation.

Our aim is to bypass the limitations of current metric embedding methods by designing approximation algorithms that given some input space $\mathcal{M}$, they compute a small subset of points to delete, and an embedding of the residual space into some desired host space.


© Karine Chubarian and Anastasios Sidiropoulos;
licensed under Creative Commons License CC-BY



1.1 Our contribution

We now formally define outlier embeddings and state our main result. Let $\mathcal{M} = (X, \rho)$, $\mathcal{M}' = (X', \rho')$ be metric spaces. An injection $f : X \to X'$ is called an *embedding*. Given an embedding f, its *distortion* is defined as

$$\mathsf{distortion}(f) = \sup_{x \neq y \in X} \frac{\rho'(f(x), f(y))}{\rho(x, y)} \cdot \sup_{x' \neq y' \in X} \frac{\rho(x', y')}{\rho'(f(x'), f(y'))}.$$

We also refer to this notion of distortion as *multiplicative distortion*. An embedding is *bi-Lipschitz* if its distortion is bounded. When $\mathcal{M}' = (\mathbb{R}, \ell_2)$ then we say that $\mathcal{M}$ admits an embedding into the line. If $\mathsf{distortion}(f) \leq c$, then we say that f is a *c-embedding*. We use the following definition for outlier embeddings (see also [19]).

A metric space $\mathcal{M} = (X, \rho)$ *admits a* (k, c)*-embedding* into another metric space $\mathcal{M}' = (X', \rho')$ for some $c \geq 1$, $k \geq 0$ if there exists $K \subseteq X$, with $|K| \leq k$, and $f : X \setminus K \to X'$, with $\mathsf{distortion}(f) \leq c$. We say that such $K \subseteq X$ is an *outlier set* (w.r.t. f).

In the present work, we focus on the case where the input metric space is the shortest-path metric of an unweighted graph, and the host space is the real line. This setting, but without outliers, has been studied extensively in the literature (see Section 1.2 for a more detailed discussion). The shortest-path metrics of unweighted graphs arise naturally in applications, for example, when considering the k-NN graph of a point set; that is, by taking the set of vertices to be a set of samples from some unknown manifold, and the edge set to be all pairs $\{u, v\}$, where u is one of the k nearest neighbors of v. Moreover, the case of embedding into the real line is a prototypical mathematical model for the problem of discovering 1-dimensional structure in a metrical data set.

The following summarizes the main result of this paper.

▶ **Theorem 1.** *Let G be a graph, $k \geq 0$, $c \geq 1$. There exists a polynomial-time algorithm which given G, k, and c, terminates with exactly one of the following outcomes:*
(1) *Correctly decides that G does not admit a (k, c)-embedding into the line.*
(2) *Computes a $(O(c^6 k \log^{5/2} n), O(c^{13}))$-embedding of G into the line.*

1.2 Related work

Low-distortion metric embeddings have been studied extensively within mathematics and computer science. We refer the reader to [14] for a detailed exposition of the work that is of main interest for computer science. Here, we discuss some results relevant to our work.

Approximation algorithms. The problem of computing an embedding of some input metric space $\mathcal{M}$ into some host space $\mathcal{M}'$ with approximately minimum distortion has received a lot of attention. Most positive results are concerned with the case where $\mathcal{M}'$ is the line, or, more generally, some 1-dimensional space. Specifically, Bădoiu et al. [6] obtained an algorithm which given an unweighted graph that admits a c-embedding into the line, computes a $O(c^3)$-embedding into the line. Approximation algorithms have also been obtained by Bădoiu et al. [5] for the case where the input is a weighted tree, and by Nayyeri and Raichel [17] for the case where the input is a general metric space.

Approximation algorithms for embedding into more general 1-dimensional spaces have also been considered. Bădoiu et al. [3] consider the case where the host space is a tree, Chepoi et al. [8] consider the case where the host space is an outerplanar graph, and Nayyeri and Raichel [18] generalize this to the case where the host space is a graph of bounded treewidth. Carpenter et al. [7] obtain an approximation algorithm for embedding unweighted graphs into subdivisions of any fixed "pattern" graph H (embedding into the line corresponds to H being a single edge, while embedding into a cycle is the case where H is a triangle).

The case of higher-dimensional host spaces appears to be significantly more challenging. The only positive results are an approximation algorithm for embedding finite subsets of the 2-sphere into $\mathbb{R}^2$ [6], and approximation algorithms for embedding ultrametrics into $\mathbb{R}^d$ [4, 10]. On the negative side, it is shown that for any $d \geq 1$, the problem of embedding into d-dimensional Euclidean space with minimum distortion is hard to approximate within a factor of $n^{\alpha/d}$, for some constant $\alpha > 0$ (the case $d = 1$ is due to [5] and $d \geq 2$ is due to [16]).

FPT algorithms. The problem of computing an embedding into the line parameterized by the optimal distortion has also been considered. Fellows et al. [12] gave an FPT algorithm for embedding unweighted graphs into the line. A nearly-matching lower bound on the running time (assuming ETH) was obtained by Lokshtanov et al. [15]. FTP algorithms for embedding unweighted graphs into subdivisions of an arbitrary fixed pattern graph H have also been obtained by Carpenter et al. [7].

Outlier embeddings. The problem of computing outlier embeddings was introduced by Sidiropoulos et al. [19]. They considered the case of embedding into d-dimensional Euclidean space, and into trees. The main difference with our work is that [19] deals with the case of *additive* distortion, while we are concerned with multiplicative distortion. As a result, the results in [19] are incomparable to ours. We remark, however, that the case of mutliplicative distortion is known to be significantly more challenging. To the best of our knowledge, our result is the first non-trivial upper bound for computing outlier embeddings minimizing the multiplicative distortion.

1.3 High-level overview of the algorithm

We now give an informal description of our algorithm, highlighting the main technical challenges. The input consists of an undirected graph G and some $k \geq 0$, $c \geq 1$. The algorithm either correctly decides that there exists no (k, c)-embedding of G into the line, or outputs a (k', c')-embedding of G into the line, for some $k' = \mathsf{poly}(k, c, \log n)$, $c' = \mathsf{poly}(c)$.

The crux of the algorithm is to identify and remove three "obsrtuctions" for low-distortion embeddability into the line. These three obstructions are *regions of high density*, *large metrical cycles* and *large metrical tripods*. We next discuss the steps used to handle each one of these obstructions, and describe how all the steps are combined in the final algorithm.

Obstruction 1: Reducing the density. The *density* of a graph is defined to be

$$\Delta(G) = \max_{v \in V(G), R \in \mathbb{N}} \frac{|\mathsf{Ball}_G(v, R)| - 1}{2R}.$$

It is known that the density of any graph that admits a c-embedding into the line is $O(c)$ [6]. Therefore, if G admits a (k, c)-embedding, then there must exist some set of at most k vertices, whose deletion leaves a graph with density $O(c)$. We observe that the density of a graph is a hereditary property, meaning that for any $H \subseteq G$, we have $\Delta(H) \leq \Delta(G)$. This leads to a following recursive procedure: if the density is higher than $O(c)$, we compute a balanced vertex separator $X \subseteq V(G)$, and recurse on $G \setminus X$. We set

$$K_{\mathsf{density}} := \bigcup_{\text{all separators } X} X.$$

Let us also denote $G \setminus K_{\mathsf{density}}$ as G'. It is immediate that $\Delta(G') = O(c)$, and we show that $|K_{\mathsf{density}}| = \mathsf{poly}(k, c, \log n)$.

Obstruction 2: Eliminating large metrical cycles. It is known that any embedding of the n-cycle into the line must incur distortion $\Omega(n)$ [6]. More generally, it is possible to define an obstruction, which we refer to as a *metrical cycle*, and which contains cycles as a special case, but allows for more general shortest-path distances (see Figure 1). We show how to delete a small number of vertices so that the resulting graph does not contain any large metrical cycles, and then we find a low-distortion embedding into some forest.

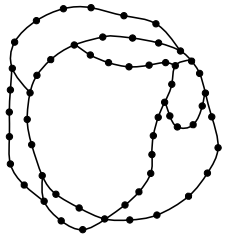


Figure 1 Example of a large metrical cycle.

We now briefly describe the procedure for eliminating large metrical cycles. We start by computing a $\mathsf{poly}(c)$-net N in G'. We then find a Voronoi partition $\mathcal{P}$ centered at N: for any vertex $v \in G'$, we assign v to a cluster centered at its nearest neighbour $y \in N$ (we break ties to ensure connectivity). Let H be the minor of G obtained by contracting each cluster to its center $y \in N$. We compute an approximate minimum feedback vertex set Y in H. We set

$$K_{\mathsf{forest}} := \bigcup_{x \in Y} \mathcal{P}(x),$$

and $G'' = G' \setminus K_{\mathsf{forest}}$. Note that the low density of G' ensures that $|K_{\mathsf{forest}}|$ is small. Furthermore, we show that G'' admits a low-distortion embedding into a forest.

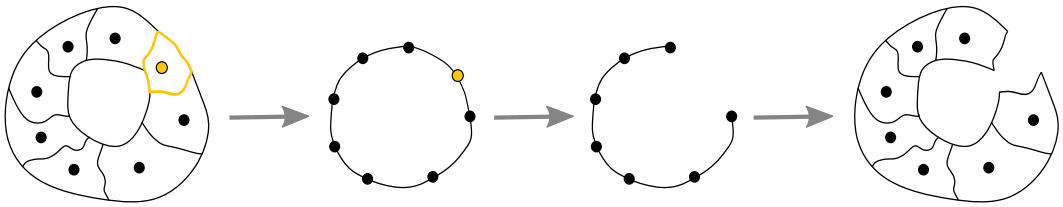


Figure 2 Elimination of large metrical cycles. From left to right: the graph G', the minor H, the forest $H \setminus Y$, and the graph G''.

Obstruction 3: Eliminating large metrical tripods. A *tripod* is a tree consisting of the union of three paths with a common endpoint; we say that a tripod is R-large if the length of each of the three paths is at least R. Any embedding of a R-large tripod into the line must incur distortion $\Omega(R)$. We show how to delete a small number of vertices so that the resulting graph does not have any subgraphs with a shortest-path metric that resembles that of a $\Omega(\mathsf{poly}(c))$-large tripod. More specifically, via a reduction to the Minimum Set Cover problem, we compute some $Z \subseteq V(H \setminus Y)$, so that the forest $H \setminus (Y \cup Z)$ does not contain any $\Omega(\mathsf{poly}(c))$-large tripods (see Figure 3). We set

$$K_{\mathsf{tripod}} := \bigcup_{w \in (H \setminus Y) \setminus Z} \mathcal{P}(w).$$

and $G'' = G' \setminus K_{\text{tripod}}$. Since the forest $H \setminus (Y \cup Z)$ does not contain any large tripods, we can show that it admits a low-distortion embedding into the line. Furthermore, we can use this embedding to also embed G'' into the line.

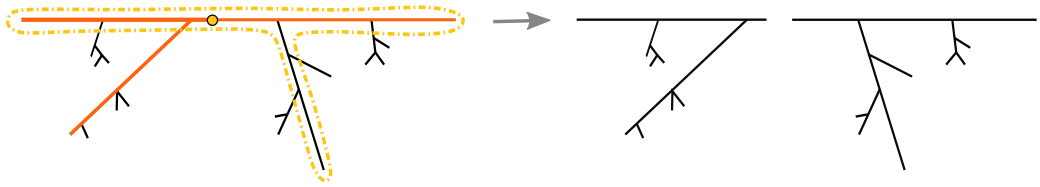


Figure 3 Elimination of a large tripod. A yellow vertex removes the red tripod and the yellow dotted tripod simultaneously.

Putting everything together. The final algorithm combines the above procedures for eliminating the three obstructions that we have identified. At each obstruction elimination step, we remove a small set of vertices. One additional complication is that, because c-embeddability into the line is not a hereditary property, this can produce a graph that does not admit a low-distortion embedding into the line. We show that this issue can be avoided by deleting a slightly larger superset of vertices, which eliminates the obstruction at hand, while maintaining the existence of a low-distortion embedding.

1.4 Organization

The rest of the paper is organized as follows. We introduce necessary notation and definitions in Section 2. In Section 3, we present our main algorithm and we state the main technical results needed. In Section 3.2 we prove a technical lemma which will be applied throughout the paper. Sections A, B, C, D elaborate on the subroutines executed by the main algorithm.

2 Preliminaries

2.1 Graphs

Given a graph G, we refer to its vertex set as $V(G)$ and to its edge set as $E(G)$. For any $C \subseteq V(G)$, we denote by $G[C]$ the subgraph of G induced on C. Let d_G denote the shortest-path distance of G; unless otherwise noted, we assume that all edges in G are undirected and have unit length.

▶ **Definition 2** (Local density). *For any $v \in V(G)$ and $R \in \mathbb{N}$, we define*

$$\Delta_G(v, R) = \frac{|\mathsf{Ball}_G(v, R)| - 1}{2R}$$

The local density *of the graph G is defined to be*

$$\Delta(G) = \max_{v \in V(G), R \in \mathbb{N}} \Delta_G(v, R).$$

▶ **Definition 3** (Tripod). *Let G be a graph, $R \geq 1$, $v, v_1, v_2, v_3 \in V(G)$, and let P_1, P_2, P_3 be paths in G, where for all $i \in [3]$, P_i is a path with endpoints v and v_i. Suppose that for all $i \neq j \in [3]$, and for all $u \in P_j$, we have $d_G(v_i, u) \geq R$. In other words, each endpoint v_i is at distance at least R from every vertex in the other two paths. Then we say that the tree $P_1 \cup P_2 \cup P_3$ is a* R-tripod *with* root v *(in G).*

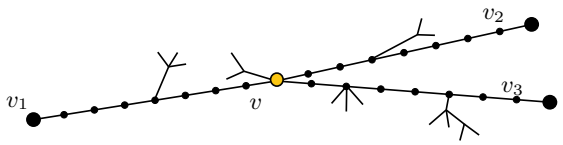


Figure 4 A tripod rooted at v with leaves v_1, v_2, v_3.

2.2 Some useful approximation results

For a graph G, a *feedback vertex set* is some $X \subseteq V(G)$, such that $G \setminus X$ is acyclic. In the Minimum Feedback Vertex Set problem we are given a graph G and the goal is to find a feedback vertex set in G of minimum cardinality. We recall the following result on approximating the Minimum Feedback Vertex Set problem.

▶ **Theorem 4** (Bafna et al. [1]). *There exists a polynomial-time 2-approximation algorithm for the Minimum Feedback Vertex Set problem.*

Given a graph G and some $\alpha \in [0, 1)$, we say that some $X \subseteq V(G)$ is a *α-balanced vertex separator* (of G) if every connected component of $G \setminus X$ has at most $\alpha \cdot |V(G)|$ vertices. We recall the following algorithmic result on computing balanced vertex separators.

▶ **Theorem 5** (Feige et al. [11]). *There exists a polynomial-time algorithm which given a graph that admits a $2/3$-balanced vertex separator of size s, outputs a $3/4$-balanced vertex separator of size at most $O(\sqrt{\log n} \cdot s)$.*

Recall that an instance to the Minimum Set Cover problem consists of some set U (the *universe*), and a set $\mathcal{C}$ of subsets of U. The goal is to find a subset of $\mathcal{C}$ of minimum cardinality that covers U.

▶ **Theorem 6** (Chvátal [9]). *There exists a polynomial-time $O(\log n)$-approximation algorithm for the Minimum Set Cover problem.*

2.3 Voronoi minors

For some metric space $\mathcal{M} = (X, \rho)$, and some $R > 0$, we say that some $N \subseteq X$ is a *R-net* of M if for any $p, q \in N$, $\rho(p, q) > R$, and $X \subseteq \bigcup_{p \in N} \mathsf{Ball}_{\mathcal{M}}(p, R)$. For a graph G, we say that some $N \subseteq V(G)$ is a R-net of G if N is a R-net of the shortest-path metric of G.

▶ **Definition 7** (Graphical Voronoi partition). *Let G be a graph, and let $Y \subseteq V(G)$. Let $\mathcal{P}$ be a partition of $V(G)$ satisfying the following conditions:*

(1) *Every cluster in $\mathcal{P}$ contains exactly one vertex in Y.*

(2) *For any $v \in V(G)$, the cluster containing v, $\mathcal{P}(v)$, also contains some nearest neighbor of v in Y.*

(3) *For any cluster $C \in \mathcal{P}$, we have that $G[C]$ is connected.*

We say that $\mathcal{P}$ is a Voronoi partition *of G* centered at *Y.*

We note the following easy fact.

▶ **Lemma 8.** *For any graph G, and $Y \subseteq V(G)$, there exists a Voronoi partition $\mathcal{P}$ of G centered at Y.*

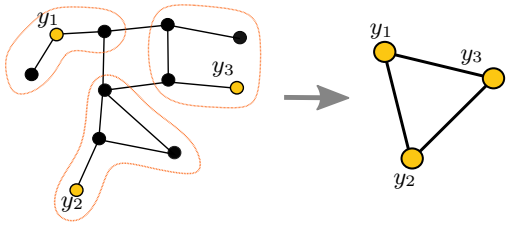


Figure 5 A Voronoi partition centred at 3-net $N = \{y_1, y_2, y_3\}$ and a corresponding 3-minor.

Proof. Construct $\mathcal{P}$ by assigning each $v \in V(G)$ to the cluster containing its nearest neighbor in Y. In order to ensure that each cluster C induces connected subgraph $G[C]$ it suffices to ensure that shortest-paths in G are unique. This can be achieved by breaking ties between different paths lexicographically (viewing paths as sequences of vertices with unique integer labels) (see also [6]). ◀

▶ **Definition 9** (R-Minor). *Let G be a graph, $R > 0$, and let N be a R-net of G. Let $\mathcal{P}$ be a Voronoi partition of G centered at N. Let H be the minor of G obtained by contracting each cluster in C in $\mathcal{P}$ into the unique net point in C. Then we say that $\mathcal{P}$ is a* R-partition *and H is a* R-minor *of G induced by $\mathcal{P}$ (see Figure 5 for an example).*

3 The Main Algorithm

In this section we present and analyze the main algorithm of the paper. For the clarity, we first state some key technical ingredients used by the algorithm. We then present the main algorithm and its analysis. The proofs of the technical ingredients appear in latter Sections.

3.1 Technical ingredients used by the main algorithm

Density reduction. The first technical ingredient used by the main algorithm is a procedure for reducing the local density of the input graph. This is summarized in Lemma 10. Its proof is given in Section A.

▶ **Lemma 10** (Density Reduction). *There exists a polynomial-time algorithm given given a graph G, $k \geq 0$, $c \geq 1$, terminates with exactly one of the following outcomes:*

(1) *Correctly decides that G does not admit a (k, c)-embedding into the line.*

(2) *Outputs some $Y \subseteq V(G)$ such that $\Delta(G \backslash Y) \leq c$, with $|Y| = O(ck \log^{3/2} n)$. In particular, if $\Delta(G) \leq c$, then the algorithm outputs $\emptyset$.*

Eliminating large metrical cycles. The next technical ingredient is a procedure for eliminating large metrical cycles. This is summarized in Lemma 11, whose proof is given in Section B.

▶ **Lemma 11** (Embedding into a forest). *There exists a polynomial-time algorithm which given a graph G, $c \geq 1$, and $k \geq 0$, terminates with exactly one of the following outcomes:*

(1) *Correctly decides that G does not admit a (k, c)-embedding into the line.*

(2) *Outputs a c-net N of G, a c-partition $\mathcal{P}$ centered at N, a c-minor H induced by $\mathcal{P}$, and some feedback vertex set X of H, with $|X| \leq 2k$.*

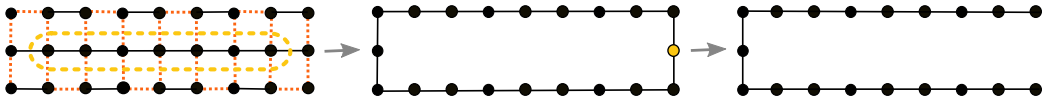


Figure 6 A $3 \times n$ grid G can be embedded into the line with distortion $O(1)$; one could follow the red dotted path on the grid an embed the vertices consequently. A yellow line depicts U. Now, if we delete a yellow vertex from $G \setminus U$, the resulting graph will be just a path.

Eliminating large metrical tripods. The next obstruction that the main algorithm needs to remove is large metrical tripods. This is done using Lemmas 12 and 13. Their proofs appear in Section C.

▶ **Lemma 12** (Tripods as obstructions to embeddability)**.** *Let G be a graph, $R \geq 1$, and let J be a R-tripod in G. Then for any c-embedding of G into the line we have $c \geq 2R$.*

▶ **Lemma 13** (Tripod elimination)**.** *There exists a polynomial-time algorithm which given a forest F, $R \geq 1$, $k \geq 0$, terminates with exactly one of the following outcomes:*

(1) *Correctly decides that there exists no $X' \subseteq V(F)$, with $|X'| \leq k$, such that $F \setminus X'$ does not contain any R-tripod as a subgraph.*

(2) *Outputs some $X' \subseteq V(F)$, with $|X'| = O(k \log n)$, such that $F \setminus X'$ does not contain any R-tripod as a subgraph.*

Embedding a tree with no large tripods into the line. Once all the obstructions have been removed, the problem is reduced to computing an embedding of a tree with no large tripods into the line. This is done using Lemma 14, whose proof appears in Section D.

▶ **Lemma 14.** *Let $R \geq 1$, and let T be a tree that does not contain any R-tripod as a subgraph. Then T admits a $O(\Delta(T) \cdot R)$-embedding into the line. Moreover, this embedding can be computed in polynomial time.*

3.2 The Repairing Lemma

The main algorithm proceeds in several steps. At each step, it uses some of the procedures described above to delete small subsets of vertices. However, because c-embeddability into the line is not a hereditary property, it is possible that the deletion of some small set of vertices destroys some candidate solution. As an illustrative example, let G be the $3 \times (n/3)$ grid. Note that G admits a $O(1)$-embedding into the line (i.e. without outliers). This embedding can be realized by consecutively traversing the columns of the grid. Let U be the set of vertices that do not lie on the outer boundary cycle of G. Then, $G \setminus U$ is the $(2n/3 + 2)$-cycle, and therefore any embedding of $G \setminus U$ into the line has distortion $\Omega(n)$. However, by removing one additional vertex from $G \setminus U$ we obtain a path, which admits a 1-embedding into the line (see Figure 6). We show that the above "repairing" process can be performed for arbitrary U. First, we prove two auxiliary statements.

▶ **Lemma 15.** *Let G be a graph, $k > 0$, $c > 1$. Assume that G admits a (k, c)-embedding into a line. Suppose G admits a (k, c)-embedding into the line realized by $f : G \setminus K \to \mathbb{R}$. Then, there exists a (k, c)-embedding f' of G into a line such that if $j > i$ then for any $v \in G_i, w \in G_j$ we have $f'(w) > f'(v)$.*

Proof. Let

$$v_1 = \arg\min_{v \in V(G) \setminus K} \{f(v)\}$$
$$v_2 = \arg\max_{v \in V(G) \setminus K} \{f(v)\}$$

and let $M = f(v_2) - f(v_1)$. Without loss of generality, we can assume that $f(v_1) = 0$ and $f(v_2) = M$ by setting $f(v) := f(v) - f(v_1)$. For each $v \in G_i$ we define $f'(v) = f(v) + 2i \cdot M$. We claim that f' and f have the same distortion. If $v, w \in G_i$ then we have

$$|f'(w) - f'(v)| = |(f(w) + 2i \cdot M) - (f(w) + 2i \cdot M)| = |f(w) - f(v)|.$$

If $v \in G_i$ and $w \in G_j$ for $i \neq j$ then the distance between them in the embedding does not contribute to the distortion.

It remains to show that $f'(w) > f'(v)$ for all $w \in G_j, v \in G_i$ with $j > i$. We have

$$f'(w) - f'(v) = f(w) - f(v) + 2(j - i)M > -M + 2M > 0$$

and the claim follows by induction. ◀

Let $G = (V(G), E(G))$ be a graph and let $f : G \to \mathbb{R}$. Consider $Z = \{v_1, \ldots, v_m\} \subseteq V(G)$ such that $f(v_1) < f(v_2) < \cdots < f(v_m)$. Then Z is *consecutive with respect to* f if for all $w \in V(G) \setminus U$ either $f(w) < f(v_1)$ or $f(v_m) < f(w)$.

▶ **Lemma 16.** *Let G be a graph, $c > 0$. Assume that G admits a $(0, c)$-embedding into the line realized by $f : G \to \mathbb{R}$. Let $Z = \{z_1, \ldots, z_m\} \subseteq V$ be consecutive with respect to f. Suppose that $f(v_m) - f(v_1) \geq c$; then Z is a vertex separator in G.*

Proof. We claim that

$$X = \{x \in V(G) \mid f(x) < f(v_1)\}$$
$$Y = \{y \in V(G) \mid f(v_k) < f(y)\}$$

are disconnected in $G \setminus Z$. Assume otherwise; then there exists $\{x, y\} \in E(G)$ with $x \in X, y \in Y$. Thus

$$|f(y) - f(x)| = f(y) - f(v_k) + f(v_k) - f(v_1) + f(v_1) - f(x) \geq c + 1 > c \cdot d_G(x, y)$$

which contradicts the distortion assumption. ◀

We can now prove the Repairing Lemma:

▶ **Lemma 17** (Repairing Lemma)**.** *Let G be a graph, $U \subset V(G)$, $k \geq 0$, $c \geq 1$. Suppose that G admits a (k, c)-embedding into a line. Then, $G \setminus U$ admits a $((2c+1)|U| + k, 4c^3 + c)$-embedding into a line.*

Proof. Let f be a (K, c)-embedding of G into the line, with $|K| = k$. Let $U' = U \cap K$, and $U'' = U \setminus K$. For any $v \in U \setminus K$, let

$$I_{\mathsf{inner}}(v) = \mathsf{Ball}_{\mathbb{R}}(f(v), c),$$
$$I_{\mathsf{outer}}(v) = \mathsf{Ball}_{\mathbb{R}}(f(v), 2c^2) \setminus I_{\mathsf{inner}}(v),$$
$$V_{\mathsf{inner}}(v) = \{u \in V(G) \setminus K : f(u) \in I_{\mathsf{inner}}(v)\}$$
$$V_{\mathsf{outer}}(v) = \{u \in V(G) \setminus K : f(u) \in I_{\mathsf{outer}}(v)\}$$

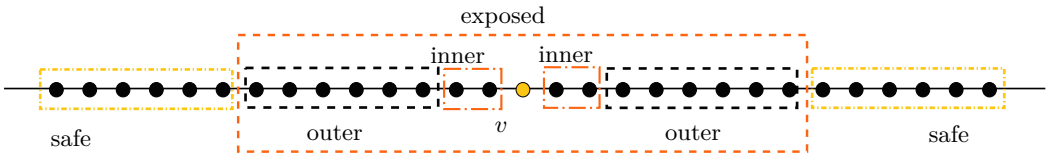


Figure 7 Inner, outer, safe and exposed vertices with respect to v for $c = 2$.

Let also

$$I_{\mathsf{inner}} = \bigcup_{v \in U \setminus K} I_{\mathsf{inner}}(v)$$

$$I_{\mathsf{outer}} = \left(\bigcup_{v \in U \setminus K} I_{\mathsf{outer}}(v) \right) \setminus I_{\mathsf{inner}}$$

$$V_{\mathsf{inner}} = \{u \in V(G) \setminus K : f(u) \in I_{\mathsf{inner}}\},$$

$$V_{\mathsf{outer}} = \{u \in V(G) \setminus K : f(u) \in I_{\mathsf{outer}}\},$$

$$V_{\mathsf{exposed}} = V_{\mathsf{inner}} \cup V_{\mathsf{outer}},$$

$$V_{\mathsf{safe}} = V(G) \setminus V_{\mathsf{exposed}}.$$

We can now define

$$K' = K \cup V_{\mathsf{inner}}.$$

Since the minimum distance in G is one, and f is non-contracting, it follows that

$$|K'| \leq |K| + (2c+1)|U|.$$

Let $c' = (4c^3 + c)$. It remains to construct any (K', c')-embedding f'. By lemma 15 it is enough to construct a c'-embedding for each connected component of $G \setminus K'$

We may thus focus on any connected component C of $G \setminus K'$. Let $f' = (4c^2+1) \cdot f|_C$ (that is, f' is the restriction of f on C scaled by a factor of $4c^2+1$). It suffices to show that f' is a $(4c^3 + c)$-embedding of C.

If there exist $v \in U \setminus K$, and $u \in C$ such that $f(v) < f(u)$, then we set

$$z_L = \operatorname*{arg\,max}_{v \in U \setminus K : \forall u \in C, f(z_L) < f(u)} \{f(v)\},$$

Similarly, if there exist $v \in U \setminus K$, and $u \in C$ such that $f(v) > f(u)$, then we set

$$z_R = \operatorname*{arg\,min}_{v \in U \setminus K : \forall u \in C, f(z_R) > f(u)} \{f(v)\}.$$

Let $u, v \in V(C)$. We first bound the expansion of f'. Since $K \subset K'$, it follows what $d_{G \setminus K}(u,v) \leq d_{G \setminus K'}(u,v)$, and thus

$$\begin{aligned} |f'(u) - f'(v)| = (4c^2+1) \cdot |f(u) - f(v)| &\leq (4c^3 + c) \cdot d_{G \setminus K}(u,v) \\ &\leq (4c^3 + c) \cdot d_{G \setminus K'}(u,v). \end{aligned} \tag{1}$$

It remains to show that f' is non-contractive. Let P be the shortest path between u and v in $G \setminus K$. Let us first assume that $u, v \in V_{\mathsf{safe}}$; we will consider the general case later. If z_L is defined and $P \cap V_{\mathsf{outer}}(z_L)$, we first construct a new path P' that avoids $V_{\mathsf{outer}}(z_L)$, as

follows. When traversing P starting from u, let u_1 be the last vertex before visiting $V_{\mathsf{outer}}(z_L)$ for the first time; let also u_2 be the first vertex visited immediately after leaving $V_{\mathsf{outer}}(z_L)$ for the last time.

Since the expansion of f is at most c, it follows that

$$f(u_1) \in (f(z_L) + 2c^2, f(z_L) + 2c^2 + c],$$
$$f(u_2) \in (f(z_L) + 2c^2, f(z_L) + 2c^2 + c],$$

and thus

$$d_{G\setminus K}(u_1, u_2) \leq |f(u_1) - f(u_2)| \leq c. \tag{2}$$

Let W be the shortest path between u_1 and u_2 in $G \setminus K$. Since every edge of W is stretched by at most a factor of c in f, it follows by (2) that W cannot enter $V_{\mathsf{inner}}(z_L)$, and thus $W \subseteq G \setminus K'$. Therefore $d_{G\setminus K'}(u_1, u_2) = d_{G\setminus K}(u_1, u_2) \leq c$. We can replace P be the path $P' := P[u, u_1] \circ W \circ P[u_2, v]$, which does not intersect $v_{\mathsf{outer}}(z_L)$. We obtain that $\mathsf{length}(P') = \mathsf{length}(P[u, u_1]) + \mathsf{length}(W) + \mathsf{length}(P[u_2, v]) \leq c + \mathsf{length}(P) \leq c + d_{G\setminus K}(u, v)$.

Next, if z_R exists and $P' \cap V_{\mathsf{outer}}(z_R) \neq \emptyset$, then via a symmetric process we can replace P' by a new path P'' between u and v in $G \setminus K$ avoids $V_{\mathsf{outer}}(z_R) \cup V_{\mathsf{outer}}(z_L)$, with

$$\mathsf{length}(P'') \leq \mathsf{length}(P') + c \leq \mathsf{length}(P) + 2c.$$

This implies that $P' \subseteq G \setminus K'$.

We therefore obtain

$$\begin{aligned} |f'(u) - f'(v)| &= (4c^2 + 1) \cdot |f(u) - f(v)| \geq (4c^2 + 1) \cdot d_{G\setminus K}(u, v) \\ &\geq (4c^2 + 1) \cdot (d_{G\setminus K'}(u, v) - 2c) > d_{G\setminus K'}(u, v). \end{aligned} \tag{3}$$

By (1) and (3) we obtain that f is a $(4c^3 + c)$-embedding of $G \setminus K'$, as required.

It remains to consider the case where either $u \in V_{\mathsf{exposed}}$, or $v \in V_{\mathsf{exposed}}$. Let Q be a shortest path between u and v in $G \setminus K'$. If $Q \cap V_{\mathsf{safe}} = \emptyset$, then $\mathsf{length}(Q) \leq 4c^2 - 2c$, thus

$$d_{G\setminus K'}(u, v) \leq (4c^2 - 2c) \leq (4c^2 - 2c) \cdot d_{G\setminus K'}(u, v),$$

which implies that f' is non-contractive, as required. We may therefore assume for the remainder of the proof that $Q \cap V_{\mathsf{safe}} \neq \emptyset$. If $u \in V_{\mathsf{exposed}}$, then we may assume w.l.o.g. that $u \in V_{\mathsf{outer}}(z_L)$. When traversing P starting from u, let u_1 be the first vertex visited immediately after leaving $V_{\mathsf{outer}}(z_L)$. When traversing Q starting from u, let u_2 be the first vertex visited in V_{safe}. By an argument identical to the one used in the previous case, we can obtain a new path between u and v, given by $Q[u, u_2] \circ W \circ P[u_1, v]$, where $\mathsf{length}(Q[u, u_2]) \leq 2c^2 - c$ (since all vertices in $Q[u, u_2]$ except the last one are contained in the rightmost segment of $V_{\mathsf{outer}}(z_L)$), $W \subseteq G \setminus K'$, and $\mathsf{length}(W) \leq c$ (as in the previous case). We thus obtain a path of length at most $d_{G\setminus K}(u, v) + 2c^2$. If $v \in V_{\mathsf{exposed}}$, we repeat the above process after exchanging u and v. We thus arrive at a path between u and v of length at most $d_{G\setminus K}(u, v) + 4c^2 \leq (4c^2 + 1) \cdot d_{G\setminus K}$, which does not intersect V_{inner}, and thus it is contained in $G \setminus K'$. It follows that f' is non-contractive, and thus a $(4c^3 + c)$-embedding, which concludes the proof. ◀

3.3 The algorithm

Given the technical ingredients presented above, we are now ready to describe our main algorithm. Recall that the input consists of a graph G, and $k \geq 0$, $c \geq 1$. The algorithm proceeds in the following steps.

Step 1: Density reduction. Using the algorithm from Lemma 10 we can either correctly decide that G does not admit a (k, c)-embedding into the line, in which case we terminate, or we compute some $X_{\mathsf{density}} \subseteq V(G)$, with $|X_{\mathsf{density}}| \leq O(ck \log^{3/2} n)$, such that $\Delta(G \setminus X_{\mathsf{density}}) \leq c$.

Step 2: Cycle elimination. Let $k' = (2c+1)|X_{\mathsf{density}}| + k$ and $c' = 4c^3 + c$. Using the algorithm from Lemma 11 we either correctly decide that G' does not admits a (k', c')-embedding into the line, or we compute a c'-net N of G', a c'-partition $\mathcal{P}$ centered at N, a c'-minor H induced by $\mathcal{P}$, and some feedback vertex set Y_{forest} of H, with $|Y_{\mathsf{forest}}| \leq 2k'$. If G' does not admit a (k', c')-embedding into the line, then we terminate by deciding that G does not admit a (k, c)-embedding into the line.

Step 3: Tripod elimination. Let $F = H \setminus Y_{\mathsf{forest}}$, and recall that Y_{forest} is a feedback vertex set for H, and thus F is a forest. Using the algorithm from Lemma 13, in polynomial time, we either decide that there exists no $Y_{\mathsf{tripod}} \subseteq V(F)$, with $|Y_{\mathsf{tripod}}| \leq k'$, such that $F \setminus T_{\mathsf{tripod}}$ does not contain any $(c'/2+1)$-tripod, in which case we terminate deciding that G does not admit a (k, c)-embedding into the line, or we compute some $Y_{\mathsf{tripod}} \subseteq V(F)$, with $|Y_{\mathsf{tripod}}| = O(k \log n)$, such that $F \setminus Y_{\mathsf{tripod}}$ does not contain any $(c'/2+1)$-tripods.

Step 4: Embedding into a forest. Let $F' = F \setminus Y_{\mathsf{tripod}}$. Let

$$X_{\mathsf{forest}} = \bigcup_{v \in Y_{\mathsf{forest}}} \mathcal{P}(v),$$

$$X_{\mathsf{tripod}} = \bigcup_{v \in Y_{\mathsf{tripod}}} \mathcal{P}(v),$$

and

$$K = X_{\mathsf{density}} \cup X_{\mathsf{forest}} \cup X_{\mathsf{tripod}}.$$

Let F'' be the forest obtained from F' as follows. Initially, we set $F'' := F'$. For each $v \in V(G) \setminus K$, let $u(v)$ be the unique vertex in $N \cap \mathcal{P}(v)$; we add v to F'' as a leaf attached to $u(v)$. This completes the construction of the forest F''.

Step 5: Embedding into the line. Finally, we compute an embedding f of F'' into the line using the algorithm from Theorem 14. We output the embedding $\varphi := 2c'c \cdot f$ (that is, f scalled by a factor of $2c'c$).

3.4 Analysis of the main algorithm

We now analyze the main algorithm presented above. First, we state some auxiliary properties of c-minors and c-partitions.

▶ **Lemma 18.** *Let G be a graph, $R \geq 1$. Let N be a R-net of G, $\mathcal{P}$ a corresponding R-partition and H a R-minor G induced by $\mathcal{P}$. Then for any $Y \subseteq V(H)$ all of the following hold:*

(1) *$N' := N \setminus Y$ is a R-net in $G' := G \setminus (\cup_{v \in Y} \mathcal{P}(v))$*

(2) *$\mathcal{P}' := \mathcal{P} \setminus (\cup_{v \in Y} \{\mathcal{P}(v)\})$ is the R-partition of G' centered at N'*

(3) *$H' := H \setminus Y$ is the R-minor of G' induced by $\mathcal{P}'$.*

Proof. We first show (1). Since by deleting vertices the shortest-path distances cannot increase, we have that for all $u, v \in N'$, $d_{G'}(u, v) \geq d_G(u, v) > R$. It thus remains to show that for any $x \in V(G')$ there exists $v \in N'$ such that $d_{G'}(x, v) \leq R$. Consider an arbitrary

$x \in V(G')$. Let $v \in N$ be such that $x \in \mathcal{P}(v)$. Since the shortest path between v and x in G is contained in $\mathcal{P}(v)$, it follows that

$$d_{G'}(x,v) \leq d_{G'[\mathcal{P}(v)]}(x,v) = d_{G[\mathcal{P}(v)]}(x,v) = d_G(x,v) \leq c,$$

which implies that N' is a c-net of G'.

Next, we show (2). Since for all $v \in N'$, we have $\mathcal{P}'(v) = \mathcal{P}(v)$, it follows that $\mathcal{P}'$ is a partition of $V(G')$. Since by (1) N' is a R-net of G', and for all $v \in N'$, and for all $x \in \mathcal{P}'(v)$ we have $d_{G'}(v,x) \leq R$, it follows that $\mathcal{P}'$ is a R-partition of G' centered at N'.

Finally, we show (3). Let $\widetilde{H}$ be the R-minor of G' induced by $\mathcal{P}'$. We prove that $V(H') = V(\widetilde{H})$ and $E(H') = E(\widetilde{H})$. For the first equality, observe that

$$V(H') = V(H \setminus Y) = N \setminus Y = N' = V(\widetilde{H}).$$

It remains to show that $E(H') = E(\widetilde{H})$. Consider an arbitrary $\{u,v\} \in E(H')$. Since $H' \subseteq H$ we have that $\{u,v\} \in E(H)$. Then there must exist a path $P \subseteq G$ between u, v with $P \subseteq \mathcal{P}(u) \cup \mathcal{P}(v)$. Since $u,v \in V(H \setminus Y) = N'$ we have that $\mathcal{P}(u) = \mathcal{P}'(u)$, $\mathcal{P}(v) = \mathcal{P}'(v)$. Thus, $P \subseteq \mathcal{P}'(u) \cup \mathcal{P}'(v)$ which yields $\{u,v\} \in E(\widetilde{H})$. Now consider an arbitrary $\{u,v\} \in E(\widetilde{H})$; it induces a path $Q \subseteq G'$ between u, v such that $Q \subseteq \mathcal{P}'(u) \cup \mathcal{P}'(v)$. Since $\mathcal{P}'(u) = \mathcal{P}(u)$, $\mathcal{P}'(v) = \mathcal{P}(v)$ we obtain $\{u,v\} \in E(H)$. Then from $u,v \in N' = N \setminus Y$ we have $\{u,v\} \in E(H \setminus Y) = E(H')$ which concludes the proof. ◀

▶ **Lemma 19.** *Let G be a graph and let $R > 0$. Let N be c-net of G, $\mathcal{P}$ a c-partition centered at N, and H a R-minor induced by $\mathcal{P}$. Then for any $u,v \in N$ we have $d_H(u,v) \leq d_G(u,v)$.*

Proof. Let $P \subseteq G$ be a shortest path between u,v and let $J := \{w \in N : P \cap \mathcal{P}(w) \neq \emptyset\}$. Let $Q \subseteq H$ be a shortest path between u,v. We claim that

$$\mathsf{length}(Q) \leq |J| - 1 \leq \mathsf{length}(P).$$

Assume for contradiction that $\mathsf{length}(Q) > |J| - 1$. Consider arbitrary $\{x_1,x_2\} \in E(P)$ such that $x_1 \in \mathcal{P}(w_1), x_2 \in \mathcal{P}(w_2)$ for $w_1 \neq w_2$; hence $\{w_1,w_2\} \in E(H)$. Therefore, P induces a walk $W \subseteq H$ such that $v,u \in V(W)$. Hence, there is a path $Q' \subseteq W$ such that $v,u \in V(Q')$; note that $\mathsf{length}(Q) \leq |V(W)| - 1 = |J| - 1$. Thus,

$$\mathsf{length}(Q') \leq |J| - 1 < \mathsf{length}(Q) = d_H(v,u).$$

which gives a contradiction, and concludes the proof. ◀

We now have all the necessary ingredients in place to prove Theorem 1, which is the main result of this paper.

Proof of Theorem 1. We analyze the algorithm presented above. By Lemma 10, if we terminate at Step 1, then we correctly decide that G does not admit a (k,c)-embedding. Otherwise, by Lemma 17, it follows that if G admits a (k,c)-embedding into the line, then $G' = G \setminus X_{\mathsf{density}}$ admits a (k',c')-embedding into the line, with $k' = (2c+1)|X_{\mathsf{density}}| + k = O(c^2 k \log^{3/2} n))$ and $c' = 4c^3 + c$.

By Lemma 11, if we decide that G' does not admit a (k',c')-embedding into the line, then, by the above discussion, this certifies that G does not admit a (k,c)-embedding into the line; we can thus correctly decide this fact in Step 2.

Suppose that G' admits a (k',c')-embedding into the line. Thus, there exists some $K' \subseteq V(G')$, with $|K'| \leq k'$, such that $G' \setminus K'$ admits a c'-embedding into the line. Let J be the set of all $v \in N$ such that the Voronoi cell of v intersects K', that is $J = \{v \in N : K' \cap \mathcal{P}(v) \neq \emptyset\}$.

We claim that $F \setminus J$ does not contain any $(3c'/2+1)$-tripod. For the sake of contradiction, suppose that $F \setminus J$ contains some $(3c'/2+1)$-tripod $T = P_1 \cup P_2 \cup P_3$, where P_1, P_2, P_3 are three paths sharing a root r. For any $i \in [3]$ let z_i be the endpoint of P_i other than r. Then for any $i \in [3]$ there exists a path Q_i in $G' \setminus K'$ between r and z_i. We claim that for all $i \neq j \in [3]$, for all $u \in V(Q_j)$, we have $d_{G \setminus K}(z_i, u) \geq c'/2 + 1$. By Lemma 18, $F \setminus J$ is a c'-minor of $G' \setminus K'$ with respect to the Voronoi partition $\mathcal{P}_J$ with $\mathcal{P}(w) = \mathcal{P}_J(w)$ for all $w \in V(F \setminus J)$. Let w' be such that $u \in \mathcal{P}_J(w')$. By Lemma 19 obtain

$$\begin{aligned}
d_{G' \setminus K'}(z_i, u) &\geq d_{G' \setminus K'}(z_i, w') - d_{G' \setminus K'}(w', u) && \text{(by the triangle inequality)} \\
&\geq d_{G' \setminus K'}(z_i, w') - c' && \text{(since } u \in \mathcal{P}_J(w')\text{)} \\
&\geq d_{F \setminus J}(z_i, w) - c' && \text{(by Lemma 19)} \\
&\geq 3c'/2 + 1 - c' && \text{(since } T \text{ is a } (3c'/2+1)\text{-tripod)} \\
&= c'/2 + 1.
\end{aligned}$$

Therefore, by Lemma 12 we conclude that $G' \setminus K'$ does not admit a c'-embedding into the line, which is a contradiction. Therefore, we have established that if G' admits a (k, c)-embedding into the line, then there exists some $J \setminus V(F)$, with $|J| \leq k'$, such that $F \setminus J$ does not contain any $(3c'/2+1)$-tripods.

Therefore, in Step 3, if we do not find a set Y_{tripod} of the desired size, then we correctly decide that G does not admit a (k, c)-embedding into the line.

Next consider the case where in Step 3 we compute a set Y_{tripod} of the desired size. Since F' does not contain any $(3c'/2+1)$-tripods, it follows by the construction of F'', that F'' does not contain any $(3c'/2+3)$-tripods (since every leaf in F becomes the center of a star in F'). Moreover, we have $\Delta(F'') \leq \Delta(F') \cdot O(c'\Delta(G'))$, since every vertex in F'' corresponds to a star that contains the vertices of a Voronoi cell in G', and every such cell has size at most $O(c'\Delta(G'))$. Thus, by Lemma 14 we compute a c''-embedding of F'' into the line, where $c'' = O(\Delta(F'')c') = O(\Delta(F')c^3\Delta(G')) = O(\Delta(F)c^3\Delta(G)) = O(\Delta(H)c^4)$, since $\Delta(\Gamma_1) \leq \Delta(\Gamma_2)$ for all $\Gamma_1 \subset \Gamma_2$. Moreover we have $\Delta(H) \leq \Delta(G') \cdot O(c' \cdot \Delta(G')) = O(c^5)$, since every vertex in H corresponds to a Voronoi cell consisting of at most $O(c' \cdot \Delta(G'))$ vertices. Therefore $c'' = O(c^9)$, and thus we have obtained a $O(c^9)$-embedding f of F'' into the line. Note that since $V(F'') = V(G \setminus K)$, it follows that f is also a (κ, σ)-embedding of G into the line, where $\kappa = |K|$, for some $\sigma \geq 1$.

It remains to bound κ and σ. We have

$$\kappa = |X_{\mathsf{density}}| + |X_{\mathsf{forest}}| + |X_{\mathsf{tripod}}|.$$

Since G admits a (k, c)-embedding into the line, it follows from Lemma 10 that

$$|X_{\mathsf{density}}| = O(ck \log^{3/2} n).$$

Moreover, $\Delta(G \setminus X_{\mathsf{density}}) \leq c$, thus for any $\widetilde{c}$-partition $\mathcal{P}$ induced by an arbitrary $\widetilde{c}$-net N of $G \setminus X_{\mathsf{density}}$, and any $v \in N$, we have

$$|\mathcal{P}(v)| = O(\widetilde{c} \cdot \Delta(G \setminus X_{\mathsf{density}})) = O(\widetilde{c} \cdot c).$$

Therefore, using Lemma 11 with $\widetilde{c} := c'$ in the Step 3 we obtain

$$|X_{\mathsf{forest}}| = O(c' \cdot c) \cdot 2k' = O((4c^3 + c) \cdot c \cdot (c^2 k \log^{3/2} n)) = O(c^6 k \log^{3/2} n).$$

Similarly, from Lemma 13, we have

$$|X_{\mathsf{tripod}}| = O(c' \cdot c)O(k' \log n) = O((4c^3 + c) \cdot c) \cdot O(c^2 k \log^{3/2} n) \log n) = O(c^6 k \log^{5/2} n),$$

which implies that

$$\kappa = O(ck \log^{3/2} n) + O(c^6 k \log^{3/2} n) + O(c^6 k \log^{5/2} n) = O(c^6 k \log^{5/2} n).$$

To find σ, we show that $G \setminus K$ admits a $O(c^4)$-embedding ι into F'' with $\iota(v) = v$ for all $v \in G \setminus K$. By Lemma 18 F' is a c'-minor of $G' \setminus (X_{\mathsf{forest}} \cup X_{\mathsf{tripod}}) = G \setminus K$ with respect to the partition $\mathcal{P}' := \mathcal{P} \setminus (\cup_{v \in Y_{\mathsf{forest}} \cup Y_{\mathsf{tripod}}} \mathcal{P}(v))$. Consider arbitrary $x_1, x_2 \in V(G \setminus K)$ and let $v_1, v_2 \in V(F')$ be such that $x_1 \in \mathcal{P}'(v_1)$, $x_2 \in \mathcal{P}'(v_2)$. Let Q be the unique v_1-v_2 path in F'. We use Q to construct a v_1-v_2 path P in $G \setminus K$, with

$$\mathsf{length}(Q) \leq \mathsf{length}(P) \leq 2c'c \cdot \mathsf{length}(Q).$$

Since F' is a c'-minor of $G \setminus K$, for any $\{w_1, w_2\} \in E(Q)$ there is $\{z_1, z_2\} \in E(G \setminus K)$ with $z_i \in \mathcal{P}'(w_i)$ for $i \in [2]$. Moreover, for any $w \in V(Q)$ the corresponding $\mathcal{P}'(w)$ is a connected subgraph such that $|V(\mathcal{P}'(w_i))| \leq 2c'\Delta(G \setminus K) + 1 = 2c'c + 1$. Thus, Q induces a walk $W \subseteq G \setminus K$ with $|V(W)| \leq 2c'c \cdot \mathsf{length}(Q)$ and $v_1, v_2 \in W$. It follows that there is a v_1-v_2 path P in W, such that

$$\mathsf{length}(P) \leq 2c'c \cdot \mathsf{length}(Q).$$

Note that since Q is the v_1-v_2 shortest path in F', we obtain

$$\mathsf{length}(P) \leq 2c'c \cdot d_{F'}(v_1, v_2) = 2c'c \cdot d_{F''}(v_1, v_2),$$

where the last equality follows from the construction of F''.

We claim that ι has contraction $O(c^4)$. By construction of F'' we have that $d_{F''}(x_i, v_i) = 1$ thus

$$d_{G \setminus K}(x_i, v_i) \leq c' \leq c' d_{F''}(x_i, v_i).$$

Therefore, we have that

$$\begin{aligned} d_{G \setminus K}(x_1, x_2) &\leq d_{G \setminus K}(x_1, v_1) + d_{G \setminus K}(v_1, v_2) + d_{G \setminus K}(v_2, x_2) \\ &\leq c' d_{F''}(x_1, v_1) + 2c'c \cdot \mathsf{length}(Q) + c' d_{F''}(v_2, x_2) \\ &\leq 2c'c \cdot d_{F''}(x_1, v_1) + 2c'c \cdot d_{F''}(v_1, v_2) + 2c'c \cdot d_{F''}(v_2, x_2). \end{aligned}$$

Since F'' is a tree, it follows that

$$2c'c \cdot d_{F''}(x_1, v_1) + 2c'c \cdot d_{F''}(v_1, v_2) + 2c'c \cdot d_{F''}(v_2, x_2) = 2c'c \cdot d_{F''}(x_1, x_2).$$

Since $c' = O(c^3)$, it follows that the contraction of ι is at most $O(c^4)$. Now we prove that the expansion of ι is $O(1)$. We claim that $d_{F''}(x_1, x_2) \leq d_{G \setminus K}(x_1, x_2) + 2$. By the construction of F'' we have

$$\begin{aligned} d_{F''}(x_1, x_2) &= d_{F''}(x_1, v_1) + d_{F''}(v_1, v_2) + d_{F''}(v_2, x_2) \\ &= d_{F''}(x_1, v_1) + d_{F'}(v_1, v_2) + d_{F''}(v_2, x_2) = d_{F'}(v_1, v_2) + 2. \end{aligned}$$

Since F' is a c'-minor of $G \setminus K$, by Lemma 19 we get

$$d_{F'}(v_1, v_2) + 2 \leq d_{G \setminus K}(v_1, v_2) + 2,$$

thus the expansion of ι is $O(1)$. Therefore, the distortion of ι is $O(c^4)$. Hence, we obtain that the map $\phi := f \circ \iota : G \setminus K \to \mathbb{R}^1$ has distortion $\sigma = O(c^9) \cdot O(c^4) = O(c^{13})$, which concludes the proof. ◀

References

1 Vineet Bafna, Piotr Berman, and Toshihiro Fujito. A 2-approximation algorithm for the undirected feedback vertex set problem. *SIAM Journal on Discrete Mathematics*, 12(3):289–297, 1999.

2 Karol Borsuk. Drei sätze über die n-dimensionale euklidische sphäre. *Fundamenta Mathematicae*, 20(1):177–190, 1933.

3 Mihai Bădoiu, Piotr Indyk, and Anastasios Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2007, New Orleans, Louisiana, USA, January 7-9, 2007*, pages 512–521. SIAM, 2007. URL: http://dl.acm.org/citation.cfm?id=1283383.1283438.

4 Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropou. Embedding ultrametrics into low-dimensional spaces. In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 187–196, 2006.

5 Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 225–233, 2005.

6 Mihai Bădoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Räcke, Ramamoorthi Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 119–128. Society for Industrial and Applied Mathematics, 2005.

7 Timothy Carpenter, Fedor V. Fomin, Daniel Lokshtanov, Saket Saurabh, and Anastasios Sidiropoulos. Algorithms for low-distortion embeddings into arbitrary 1-dimensional spaces. In Bettina Speckmann and Csaba D. Tóth, editors, *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, volume 99 of *LIPIcs*, pages 21:1–21:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.SoCG.2018.21.

8 Victor Chepoi, Feodor F Dragan, Ilan Newman, Yuri Rabinovich, and Yann Vaxes. Constant approximation algorithms for embedding graph metrics into trees and outerplanar graphs. *Discrete & Computational Geometry*, 47(1):187–214, 2012.

9 Vasek Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of operations research*, 4(3):233–235, 1979.

10 Mark de Berg, Krzysztof Onak, and Anastasios Sidiropoulos. Fat polygonal partitions with applications to visualization and embeddings. *arXiv preprint*, 2010. arXiv:1009.1866.

11 Uriel Feige, MohammadTaghi Hajiaghayi, and James R Lee. Improved approximation algorithms for minimum weight vertex separators. *SIAM Journal on Computing*, 38(2):629–657, 2008.

12 Michael R. Fellows, Fedor V. Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances A. Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 463–474. Springer, 2009. doi:10.1007/978-3-642-02927-1_39.

13 Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

14 Piotr Indyk, Jiří Matoušek, and Anastasios Sidiropoulos. 8: low-distortion embeddings of finite metric spaces. In *Handbook of discrete and computational geometry*, pages 211–231. Chapman and Hall/CRC, 2017.

15 Daniel Lokshtanov, Dániel Marx, and Saket Saurabh. Slightly superexponential parameterized problems. 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 760–776. SIAM, 2011. doi:10.1137/1.9781611973082.60.

16 Jiří Matoušek and Anastasios Sidiropoulos. Inapproximability for metric embeddings into r^d. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 405–413. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.21.

17 Amir Nayyeri and Benjamin Raichel. Reality distortion: Exact and approximate algorithms for embedding into the line. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pages 729–747. IEEE, 2015.

18 Amir Nayyeri and Benjamin Raichel. A treehouse with custom windows: Minimum distortion embeddings into bounded treewidth graphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 724–736. SIAM, 2017.

19 Anastasios Sidiropoulos, Dingkang Wang, and Yusu Wang. Metric embeddings with outliers. 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 670–689. SIAM, 2017. doi:10.1137/1.9781611974782.43.

A Density Reduction

A.1 The algorithm for density reduction

Let us describe the algorithm for reducing the density of a graph. The algorithm takes as input a graph G and some $k \geq 0$, $c \geq 1$, and outputs some $Y \subseteq V(G)$, such that $\Delta(G \setminus Y) \leq c$. This is summarized in Algorithm 1.

Algorithm 1 SPARSIFY.

```
procedure SPARSIFY(G, c)
    if Δ(G) ≤ c then
        return ∅
    else
        Let X be a 3/4-balanced vertex separator of G computed by Theorem 5.
        Let G_1, ..., G_t be the connected components of G \ X.
        return X ∪ ( ⋃_{i=1}^{t} SPARSIFY(G_i, c) )
```

A.2 Analysis of the algorithm for density reduction

We now analyze the algorithm described above. We first recall the following result from [6].

▶ **Lemma 20** (Bădoiu et al. [6]). *If G admits a c-embedding into the line then $\Delta(G) \leq c$.*

The following establishes the existence of small balanced separators.

▶ **Lemma 21.** *Let G be a graph such that G admits a (k, c)-embedding into the line. Let $Z \subseteq V(G)$ with $|Z| = k$ be such that $G \setminus Z$ is c-embeddable into the line. Then any $H \subseteq G$ contains a 2/3-balanced vertex separator of size at most $c + |Z \cap V(H)|$.*

Proof. Let $f : G \setminus Z \to \mathbb{R}$ be an embedding with distortion c. Let $V(H) = \{v_1, \ldots, v_h\}$, and assume w.l.o.g. that $f(v_1) < f(v_2) < \ldots < f(v_h)$. Let $X = \{v_{\lfloor h/3 \rfloor + 1}, v_{\max\{h, \lfloor h/3 \rfloor + c + 1\}}\}$. By lemma 16 we get that X is a balanced separator of $H \setminus Z = H \setminus (Z \cap V(H))$. Therefore, $W := X \cup (V(H) \cap Y)$ is a balanced separator for H, with $|W| = |X| + |Z \cap V(H)| \leq c + |Z \cap V(H)|$, as required. ◀

We are now ready to prove the main result of this Section.

Proof of Lemma 10. It is immediate that the output, Y, of the procedure SPARSIFY is such that $\Delta(G \setminus Y) \leq c$. Also, if $\Delta(G) \leq c$, the algorithm outputs $Y = \emptyset$.

It thus remains to bound $|Y|$. Fix some $K \subseteq V(G)$, with $|K| = k$, and some c-embedding f of $G \setminus K$ into the line. Consider some recursive call of procedure SPARSIFY(H, c), for some $H \subseteq G$. If $H \cap K = \emptyset$, then $H \subseteq G \setminus K$, and thus $\Delta(H) \leq \Delta(G \setminus K) \leq c$, where the last inequality follows by lemma 20. Therefore, procedure SPARSIFY computes a balanced separator, X_H, only if H intersects K. By lemma 21 and Theorem 5 it follows that

$$|X_H| \leq O\left(\sqrt{\log n} \cdot (c + |K \cap V(H)|)\right) \leq O\left(|K \cap V(H)| \cdot c \cdot \sqrt{\log n}\right).$$

We charge the vertices in X_H to the vertices in $K \cap H$; thus every vertex in $K \cap H$ receives at most $O\left(\sqrt{\log n}\right)$ units of charge. Since any two subgraphs on the same level of the recursion are disjoint, it follows that each vertex in K receives at most $O\left(c\sqrt{\log n}\right)$ units of charge per level of the recursion. Since each separator is 3/4-balanced, it follows that the depth of the recursion is at most $\log_{4/3} n$. Thus, every vertex in K receives at most $\log_{4/3} n \cdot O(c \log n) = \beta \cdot c \log^{3/2}_{4/3} n$ units of charge throughout the execution of the procedure SPARSIFY. The constant β comes from the bound on the size of the vertex separator computed by Theorem 5. Hence, if $Y > \beta \cdot kc \log^{3/2}_{4/3} n$, then we have certified that G does not admit a (k, c)-embedding into the line, which concludes the proof. ◀

B Eliminating large metrical cycles

B.1 The algorithm

The input consists of a graph G, some $c \geq 1$, and $k \geq 0$. The algorithm proceeds in steps, that are formally described below.

Algorithm for eliminating large metrical cycles:

Step 1. Compute a c-net N of G.

Step 2. Compute a Voronoi partition $\mathcal{P}$ of G centered at N, and the corresponding c-minor H of G.

Step 3. Using the algorithm from Theorem 4 compute a 2-approximate solution S to the Minimum Feedback Vertex Set problem on H. If $|S| > 2k$, then decide that G does not admit a (k, c)-embedding into the line.

B.2 Analysis

First, we prove the following statement about embeddability into a subgraph of a c-minor.

▶ **Lemma 22.** *Let G be a graph, $R > 0$, let N be a R-net in G, let $\mathcal{P}$ be a R-partition centered at N, and let H be the R-minor of G induced by $\mathcal{P}$. Let $X \subset N$, and let*

$$Y = \bigcup_{x \in X} \mathcal{P}(x).$$

Then the metric space $(N \setminus X, d_{G \setminus Y})$ admits a $(2R+1)$-embedding into $H \setminus X$. Moreover, this embedding can be computed in polynomial time.

Proof. Let $u, v \in N \setminus X$. Let Q be a u-v shortest path in $G \setminus Y$. When traversing Q starting from u let $C_1, \ldots, C_\ell$ be the sequence of clusters of $\mathcal{P}$ visited. For each $i \in [\ell]$ let q_i be the center of C_i; that is, $C_i = \mathcal{P}(q_i)$. Since for all $i \in [\ell - 1]$ there is an edge in $G \setminus Y$ between some vertex in C_i and some vertex in C_{i+1}, it follows that there also exists an edge in $H \setminus X$ between q_i and q_{i+1}. Therefore $Q' = q_1, \ldots, q_\ell$ is a path in $H \setminus X$. We thus obtain

$$d_{H\setminus X}(u,v) \leq \mathsf{length}(Q') \leq \mathsf{length}(Q) = d_{G\setminus Y}(u,v). \tag{4}$$

Let $W = w_1, \ldots, w_t$ be a u-v shortest path in $H \setminus X$. Since each cluster in $\mathcal{P}$ has radius at most R, it follows that for all $i \in [t-1]$ there exists a w_i-w_{i+1} path in $G \setminus Y$ of length at most $2R+1$. Concatenating all these paths we obtain a u-v path W' in $G \setminus Y$ of length at most $(t-1)\cdot(2R+1)$. Thus

$$d_{G\setminus Y}(u,v) \leq \mathsf{length}(W') \leq (2R+1)(t-1) = (2R+1)d_{H\setminus X}(u,v). \tag{5}$$

Combining (4) and (5) the assertion follows. ◀

We recall the Borsuk-Ulam Theorem [2].

▶ **Theorem 23** (Borsuk-Ulam Theorem [2]). *Let $d \geq 1$, and let $\mathbb{S}^d$ denote the d-dimensional sphere. Let $f : \mathbb{S}^d \to \mathbb{R}^d$ be a continuous map. Then there exists $x \in \mathbb{S}^d$, such that $f(x) = f(-x)$.*

The following is a simple consequence of Theorem 23. A similar argument is used in [6].

▶ **Lemma 24.** *Let C be a cycle and let $f : V(C) \to \mathbb{R}$ be an injective map. Then there exist $u, v, w \in V(C)$, such that $\{u,v\} \in E(C)$, and $f(u) < f(w) < f(v)$.*

Proof. Suppose that C is the n-cycle for some $n \in \mathbb{N}$. We identify the vertices in C with distinct points in $\mathbb{S}^1$, so that the points appear in the same order as in C along a clockwise traversal of $\mathbb{S}^1$. For each $\{x,y\} \in E(C)$ ther exists an arc $A_{x,y}$ in $\mathbb{S}^1$ that does not contain any other vertex in C; we extend f to $A_{x,z}$ affinely. After repeating for all edges in C, we obtain a continuous map $f : \mathbb{S}^1 \to \mathbb{R}^1$. By Theorem 23 we get that there exists $x \in \mathbb{S}^1$ with $f(x) = f(-x)$. This means that there exist two edges in C whose images in f span overlapping intervals in $\mathbb{R}^1$. Since f is injective on $V(C)$ this implies that one endpoint is contained inside the interval of the other edge, which concludes the proof. ◀

We next establish the existence of a small feedback vertex set in the minor computed by the algorithm.

▶ **Lemma 25.** *Let G be a graph, $c \geq 1$, $k \geq 0$, such that G admits a (k,c)-embedding into the line. Let H be a R-minor of G, for some $R \geq c$. Then there exists a feedback vertex set X in H with $|X| \leq k$.*

Proof. Let $\mathcal{P}$ be the R-partition of G such that H is the R-minor of G induced by $\mathcal{P}$. Since G admits a (k,c)-embedding into the line, it follows that there exists some $Y \subseteq V(G)$, with $|Y| \leq k$, such that $G \setminus Y$ admits a c-embedding f into the line.

Let X be the set of all $v \in V(H)$, such that Y intersects the cluster in $\mathcal{P}$ centered at v; that is $X = \{v \in V(H) : \mathcal{P}(v) \cap Y \neq \emptyset\}$. Since $\mathcal{P}$ is a partition, it is immediate that $|X| \leq |Y| \leq k$. It therefore remains to show that $H \setminus X$ is acyclic. Suppose, for the sake of contradiction, that $H \setminus X$ is not acyclic. Let C be a cycle in $H \setminus X$. By Lemma 24 there exist $u, v, w \in V(C)$, such that $\{u,v\} \in E(C)$, and $f(u) < f(w) < f(v)$.

Since $\{u,v\} \in E(C)$, and $C \subseteq H$, it follows that $\{u,v\} \in E(H)$. Since H is R-minor, it follows that there exists a path Q between u and v, with $Q \subseteq \mathcal{P}(u) \cup \mathcal{P}(v)$. When traversing Q starting from u let u' be the last vertex visited with $f(u') < f(w)$; let also v' be the vertex visited immediately after u'. We have $f(u') < f(w) < f(v')$.

Since H is a R-minor and $u' \notin \mathcal{P}(w)$, it follows that $d_G(w,u') \geq d_G(u,u')$. By the definition of a R-partition we have that $d_G(u,w) > c$, and therefore $d_G(u',w) > R/2$. Similarly, we obtain $d_G(v',w) > R/2$. Since f is non-contracting, we obtain $|f(u') - f(v')| = |f(u') - f(w)| + |f(w) - f(v')| \geq d(u,w) + d(w,v) > R/2 + R/2 = R \geq c$, which contradicts the fact that f has expansion at most c, and concludes the proof. ◀

We are now ready to prove the main result of this Section.

Proof of Lemma 11. By Lemma 25, either G does not admit a (k,c)-embedding into the line, or there exists $X \subseteq V(H)$, with $|X| \leq k$, such that $H \setminus X$ is acyclic. Using the algorithm from Theorem 4 we compute in Step 3 a 2-approximation $S \subseteq V(H)$ to the Minimum Feedback Vertex Set in H. Therefore, if $|S| > 2k$, then we can terminate with outcome (1), and otherwise terminate with outcome (2), which completes the proof. ◀

C Eliminating large metrical tripods

In this Section we present and analyze the procedure for eliminating large metrical tripods. We begin by showing that large tripods are an obstruction to embeddability into the line. This is summarized in Lemma 12.

Proof of Lemma 12. Let f be a non-contractive embedding of J into the line. Let v be the common endpoint of P_1, P_2, P_3. For each $i \in [3]$ let v_i be the other endpoints of P_i. We may assume w.l.o.g. (by change of indices) that $f(v_1) < f(v_2) < f(v_3)$. Let Q be the unique v_1-v_3 path in J. It follows that there exists $\{u,w\} \in E(Q)$, such that $f(u) < f(v_2) < f(w)$. This implies that $|f(u) - f(w)| = |f(u) - f(v_2)| + |f(v_2) - f(w)| \geq d_G(u,v_2) + d_G(v_2,w) \geq 2R = 2Rd_J(u,w)$. Therefore the distortion of f is at least $2R$, which concludes the proof. ◀

The above easily implies the following results, which asserts the existence of a small set of vertices whose removal eliminates all large tripods.

▶ **Lemma 26.** *Let F be a forest that admits a (k,c)-embedding into the line. Then there exists some $X \subseteq V(F)$, with $|X| \leq k$, such that $F \setminus X$ does not contain any $(c/2+1)$-tripod as a subgraph.*

Proof. Since F admits a (k,c)-embedding into the line, it follows that there exists some $X \subseteq V(F)$, with $|X| \leq k$, such that $F \setminus X$ admits a c-embedding into the line. It suffices to show that $F \setminus X$ does not contain any $(c/2+1)$-tripods. Suppose, for the sake of contradiction, that $F \setminus X$ contains some $(c/2+1)$-tripod J. Since $(V(J), d_J)$ is a submetric of $(V(F) \setminus X, d_{F \setminus X})$, it follows that J admits a c-embedding into the line, which contradicts Lemma 12, and concludes the proof. ◀

Now are now ready to prove the main result of this Section.

Proof of Lemma 13. Any tripod $T \subseteq F$ can be uniquely specified by selecting its root and its three leaves. Therefore, there are at most $O(|V(F)|^4)$ distinct tripods in F. Moreover, the set of all tripods, $\mathcal{T}$, can be enumerated in polynomial time. We form an instance of the Minimum Set Cover problem with universe $U = \mathcal{T}$. We also let

$$\mathcal{C} = \bigcup_{v \in V(F)} \{C_v\},$$

where $C_v = \{T \in \mathcal{T} : v \in V(T)\}$. It is immediate that for any $Y \subseteq V(F)$, $F \setminus Y$ contains no R-tripods iff $\bigcup_{v \in Y} C_v = U$. Therefore, computing a minimum-cardinality subset of vertices of F whose deletion removes all R-tripods, is equivalent to solving the Minimum Set Cover instance on $(U, \mathcal{C})$. The result now follows from Theorem 6. ◀

D Embedding Trees Without Large Tripods into the Line

This Section is devoted to proving Lemma 14, which asserts that any tree with no large tripods admits a low-distortion embedding into the line.

Proof of Lemma 14. Since T is a tree, we can compute in polynomial time a longest path Q in T. Let $Q = v_1, \ldots, v_t$. Let $\mathcal{P}$ be a Voronoi partition centered at $V(Q)$. Since T does not contain any R-tripod as a subgraph, it follows that for all $u \in V(T)$, there exists some $v \in V(Q)$, with $d_T(u, v) < R$. Therefore, for each $v_i \in V(Q)$, we have

$$|\mathcal{P}(v_i)| \leq |\mathsf{Ball}_T(v_i, R-1)| \leq \Delta(T) \cdot 2(R-1) + 1 \leq \Delta(T) \cdot 2R - 1.$$

By the definition of a graphical Voronoi partition we have that for all $i \in [t]$, the vertex-induced subgraph $T_i := T[\mathcal{P}(v_i)]$ is connected, and thus T_i is a subtree of T. Let W_i be a closed walk in T_i that visits all vertices in T_i, obtained by duplicating every edge (or, equivalently, the walk obtained by any traversal of T_i). Since every edge in T_i is traversed twice, we have $\mathsf{length}(T_i) = 2(|V(T_i)| - 1)$. Let $W_i = w_{i,1}, \ldots, w_{i,t_i}$.

We define the embedding $f_i : V(T_i) \to \mathbb{R}$ as follows. For each $v \in V(T_i)$, we define $f_i(v) = \min\{j \in [t_i] : v = w_{i,j}\}$.

We combine the mappings $f_1, \ldots, f_t$ into a mapping $f : V(G) \to \mathbb{R}$. Informally, this is done by translating each f_i so that for all $i \in [t-1]$, the image of f_i appears to the left of the image of f_{i+1}, and there is a gap of length $2R$ between these two images.

Formally, for each $u \in \mathcal{P}_{v_i}$, we set $f(u) = L_i + f_i(u)$, where

$$L_i = \begin{cases} 0 & \text{if } i = 0 \\ L_{i-1} + \max_{z \in \mathcal{P}(v_{i-1})}\{f_{i-1}(z)\} + 2R & \text{otherwise} \end{cases}$$

This completed the definition of the embedding f.

It remains to bound the distortion of f. For vertices that lie in the same cluster in $\mathcal{P}$, the map is non-contractive since the distance in the embedding is at least the distance in some walk W_i, which is at least the distance in T. Moreover, the expansion is upper bounded by the length of the walk, which is at most $\Delta(T) \cdot (2R-1)$.

Next, let us consider $p, q \in V(T)$ that fall in different clusters in $\mathcal{P}$. Suppose that $p \in \mathcal{P}(v_i)$, and $q \in \mathcal{P}(v_j)$, for some $i, j \in [t]$, with $i < j$. We have

$$\begin{aligned} |f(p) - f(q)| &\leq 2R(j-1) + \sum_{r=i}^{j} \mathsf{length}(W_i) \leq (j-1)2R + (j-i+1)\Delta(T) \cdot (2R-1) \\ &\leq (j-i) \cdot O(\Delta(T) \cdot R) = d_T(v_i, v_j) \cdot O(\Delta(T) \cdot R) \leq d_T(p, q) \cdot O(\Delta(T) \cdot R). \end{aligned}$$

Moreover $|f(p) - f(q)| \geq 2R(j-i) + 1 \geq 2R + (j-i) \geq d_T(p, v_i) + d_T(v_i, v_j) + d_T(v_j, q) = d_T(p, q)$.

Therefore, in all cases, f is non-contractive and has expansion at most $O(\Delta(T) \cdot R)$. ◀

Online Minimum Cost Matching with Recourse on the Line

Nicole Megow
Department for Mathematics and Computer Science, University of Bremen, Germany
nicole.megow@uni-bremen.de

Lukas Nölke
Department for Mathematics and Computer Science, University of Bremen, Germany
noelke@uni-bremen.de

Abstract

In online minimum cost matching on the line, n requests appear one by one and have to be matched immediately and irrevocably to a given set of servers, all on the real line. The goal is to minimize the sum of distances from the requests to their respective servers. Despite all research efforts, it remains an intriguing open question whether there exists an $O(1)$-competitive algorithm. The best known online algorithm by Raghvendra [29] achieves a competitive factor of $\Theta(\log n)$. This result matches a lower bound of $\Omega(\log n)$ [3] that holds for a quite large class of online algorithms, including all deterministic algorithms in the literature.

In this work, we approach the problem in a recourse model where we allow to revoke online decisions to some extent, i.e., we allow to reassign previously matched edges. We show an $O(1)$-competitive algorithm for online matching on the line with amortized recourse of $O(\log n)$. This is the first non-trivial result for min-cost bipartite matching with recourse. For so-called alternating instances, with no more than one request between two servers, we obtain a near-optimal result. We give a $(1+\varepsilon)$-competitive algorithm that reassigns any request at most $O(\varepsilon^{-1.001})$ times. This special case is interesting as the aforementioned quite general lower bound $\Omega(\log n)$ holds for such instances.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Online algorithms; Theory of computation → Graph algorithms analysis

Keywords and phrases min-cost matching in bipartite graphs, recourse, competitive analysis, online

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.37

Category APPROX

Related Version https://arxiv.org/abs/2001.03107

Funding *Nicole Megow*: Partially supported by the German Science Foundation (DFG) under contract ME 3825/1.

1 Introduction

Matching problems are among the most fundamental problems in combinatorial optimization with great importance in theory and applications. In the bipartite matching problem, we are given a complete bipartite graph $G = (R \cup S, E)$ with positive edge cost c_e for $e \in E$. Elements of R and S are called *requests* and *servers*, respectively, with $n := |R| \leq |S|$. A *matching* $M \subseteq E$ is a set of pairwise non-incident edges. A matching is called *complete* if every request in R is matched to a server in S, i.e. if it is incident to exactly one edge of M. The task is to compute a complete matching of minimum cost, where the *cost* of a matching M is $c(M) := \sum_{e \in M} c_e$. When all information is given in advance, the optimum can be computed efficiently, e.g., by using the Hungarian Method [22].

In the *online setting*, however, the set of requests is not known a priori. Requests arrive online, one by one, and have to be matched immediately and irrevocably to a unmatched server. As we cannot hope to find an optimal matching under these restrictions, we use standard


© Nicole Megow and Lukas Nölke;
licensed under Creative Commons License CC-BY

competitive analysis to evaluate the performance of algorithms. An online algorithm is α-*competitive* if it computes for any instance a matching M with $c(M) \leq \alpha \cdot \mathrm{OPT}$, where OPT is the cost of an optimal matching. For arbitrary edge costs, the competitive ratio of any online algorithm is unbounded [19,21]. For metric costs, there is a deterministic $(2n-1)$-competitive algorithm and this is optimal for deterministic online algorithms [19, 21]. A remarkable recent result by Nayyar and Raghvendra [27] is a fine-grained analysis of an online algorithm based on *t-net cost* [28] showing a competitive ratio of $O(\mu(G)\log^2 n)$, where $\mu(G)$ is the maximum ratio of minimum TSP tour and weighted diameter of a subset of G.

So far, the online matching problem has resisted all attempts for achieving an $O(1)$-competitive algorithm even for special metric spaces such as the line. In *online matching on the line* the edge costs are induced by a line metric; that is, we identify each vertex of G with a point on the real line and the cost of an edge between a request and a server equals their distance on the line. The competitive ratio of the aforementioned algorithm is then $O(\log^2 n)$, as $\mu(G) = 2$. This has been improved to $\Theta(\log n)$ [29], which is best possible for a large class of algorithms [2]. It remains a major open question whether there exists an $O(1)$-competitive online algorithm (deterministic or randomized) on the line.

In this paper, we consider online matching on the line with *recourse*. In the recourse model, we allow to change a small number of past decisions. Specifically, at any point, we may delete a set of edges $\{(r_i, s_i)\}_i$ of the current matching and rematch the requests r_i to different (free) servers. Online optimization with recourse is an alternative model to standard online optimization which has received increasing popularity recently; see, e.g., [1, 4, 6, 13, 18, 25]. Obviously, if the recourse is not limited then one can just simulate an optimal offline algorithm, and the online nature of the problem disappears. We say an algorithm requires *amortized recourse budget* β if it rematches requests at most βn times in total. The challenging question for online matching on the line is whether it is possible to maintain an $O(1)$-competitive solution with bounded recourse, i.e., with sublinear recourse budget.

Our results. We answer this question to the affirmative and give non-trivial results for the min-cost online matching problem with recourse. We show that with limited recourse, one can indeed maintain a constant competitive solution on the line.

▶ **Theorem 1.** *The online bipartite matching problem on the line admits an $O(1)$-competitive algorithm with amortized recourse budget $O(\log n)$.*

Our algorithm builds on the t-net-cost algorithm by Raghvendra [28, 29]; details follow later. It has the nice property that it interpolates between an $O(\log n)$-competitive online solution (without recourse) and an $O(1)$-approximate offline solution (with large recourse).

Furthermore, we investigate a special class of instances, called *alternating* instances, where between any two requests on the line there is at least one server. This class is interesting as the quite strong lower bound of $\Omega(\log n)$ for a large class of algorithms given in [3], that includes all deterministic online algorithms *without recourse* in literature, holds even on such instances. For alternating instances, we present a more direct and near-optimal algorithm with a scalable performance-recourse trade-off.

▶ **Theorem 2.** *For alternating instances of online matching on the line, there exists a $(1+\varepsilon)$-competitive algorithm that reassigns each request $O(\varepsilon^{-1.001})$ times.*

While the algorithm is quite simple, the proof requires a clever charging scheme that exploits the special structure of optimal solutions on alternating instances. We observe that a large number of recourse actions for a specific request involves large edges in the optimal solution elsewhere on the line.

As a byproduct we give a simple analysis of (a variant of) the algorithm in the traditional online setting without recourse. We show that it is $O(\log \Delta)$-competitive for alternating requests on the line, where Δ is the ratio between the largest and shortest request-server distance. This result compares to $\Theta(\log n)$ for the currently best known online algorithm [29].

▶ Remark. Simultaneously and independently of our work, Gupta, Krishnaswamy and Sandeep [17] obtained a similar result for online min-cost matching with recourse on the line. Their algorithm builds on the $O(n)$-competitive Permutation algorithm [19, 21] and adapts it for the recourse setting. On the line this is done by first matching edges according to Permutation and then asymmetrically applying recourse to arcs (r, s) of the current matching that overlap in a certain way. Both, their algorithm and analysis are completely different from ours. They further obtain a more general $O(\log n)$-competitive algorithm with amortized recourse $O(\log n)$ for arbitrary metrics.

Further related work. Extensive literature is devoted to online bipartite matching problems. The maximum matching variant is quite well understood. For the unweighted setting optimal deterministic and randomized algorithms with competitive ratio 2 and $e/(e-1)$ are known [20]. The weighted maximization setting does not admit bounded guarantees in general, but intensive research investigates models with additional assumptions; see, e.g., the survey [26]. The online min-cost matching problem is much less understood. It remains a wide open question whether a constant-competitive algorithm, deterministic or randomized, is possible for online min-cost matching on the line. In fact, the strongest known lower bound is $9+\varepsilon$ [10]. For a quite large class of algorithms, including all deterministic ones in the literature, there is lower bound of $\Omega(\log n)$ [3].

Randomization allows an improvement upon the best possible deterministic competitive ratio of $(2n-1)$ for metric online bipartite matching [19,21]; there is an $O(\log^2 n)$-competitive randomized algorithm [5]. On the line, no such improvement on the deterministic result by randomization is known; the competitive factor of $O(\log n)$ is the best known result for both, deterministic and randomized, algorithms [16, 29].

Interestingly, when assuming randomization in the order of request arrivals (instead of an adversarial arrival order), the natural Greedy algorithm is n-competitive [11] for general metric spaces. Furthermore, the online *t-net cost* algorithm is in this case $O(\log n)$-competitive [28] here. Very recently, Gupta et al. [14] gave an $O((\log\log\log n)^2)$-competitive algorithm in the model with online known i.i.d. arrivals.

Maintaining an online cardinality-maximal bipartite matching with recourse was studied extensively; see, e.g., [1, 6–8, 12, 30] and references therein. Bernstein et al. [6] showed recently that the 2-competitive greedy algorithm uses amortized $O(n \log^2 n)$ reassignments, leaving a small gap to the lower bound of $\Omega(n \log n)$. In contrast, for the min-cost variant, it remained a challenging question whether recourse can improve upon the competitive ratio. Even on the line, it remained open whether and how recourse can improve the bound of $O(\log n)$ [29].

The following two models address other types of matching with recourse. In a setting motivated by scheduling, several requests can be matched to the same server and the goal is to minimize the maximum number of requests assigned to a server. Gupta et al. [15] achieve an $O(1)$-competitive ratio with amortized $O(n)$ edge reassignments. A quite different two-stage robust model has been proposed recently by Matuschke et al. [24]. In a first stage one must compute a perfect matching on a given graph and in a second stage a batch of $2k$ new nodes appears which must be incorporated into the first-stage solution to maintain a low-cost matching by reassigning only few edges. For matching on the line, they give an algorithm that maintains a 10-approximate matching reassigning $2k$ edges.

Recourse in online optimization has been investigated also for other min-cost problems even though less than for maximization problems. Most notably seems the online minimum Steiner tree problem [13, 18, 23, 25]. Here, one edge reassignment per iteration suffices to maintain an $O(1)$-competitive algorithm [13], whereas the online setting without recourse admits a competitive ratio of $\Omega(\log n)$.

The recourse model has some relation to *dynamic algorithms*. Instead of minimizing the number of past decisions that are changed (recourse), the dynamic model focuses on the running time to implement this change (*update time*). A full body of research exists on maximum (weighted) bipartite matching; we refer to the nice survey in [9]. We are not aware of any results for min-cost matching.

2 Preliminaries

A path P is called *alternating* with respect to a matching M, if every other edge in P is contained in M. An alternating path is called *augmenting* with respect to M if it starts and ends at vertices are not covered by M. A common method for increasing the cardinality of an existing matching M is to augment along an augmenting path P. After augmentation, the resulting matching $\tilde{M}$ is given by the symmetric difference[1] $M \oplus P$. There may be a choice between different augmenting paths; typically, a path of minimum cost (w.r.t. some metric) is selected. Recently, Raghvendra [28] introduced the following metric. For $t > 1$, the *t-net-cost* of a path P w.r.t. a matching M is

$$\phi_t^M(P) \coloneqq t \cdot c(P \setminus M) - c(P \cap M) \;=\; t \cdot c(P \cap \tilde{M}) - c(P \cap M). \tag{1}$$

Our algorithm maintains three matchings: the *recourse matching* M, the actual output of the algorithm, and two auxiliary matchings based on (online and offline versions of) the t-net-cost algorithm [28], namely, the *offline matching* M^* and the *online matching* M'. While M^* is a near-optimal offline matching that possibly requires a large amount of recourse, M' is an online matching that is $O(\log n)$-competitive [29] but uses no recourse. We describe how M^* and M' are obtained based on the above cost function; see also [27–29]. By speaking of the matching M_i, M_i^* or M_i', we refer to the state of the respective matching after serving the i-th request including possible reassignments.

On arrival of the i-th request r_i, the offline t-net-cost algorithm constructs the offline matching M_i^* by augmenting M_{i-1}^* along an alternating path P_i of minimum t-net-cost w.r.t. M_{i-1}^*. That is, $M_i^* \coloneqq M_{i-1}^* \oplus P_i$. By definition, this path starts at r_i and ends at a free server, which we denote by s_i. While this procedure may require a large amount of recourse, the resulting matching has been shown to have bounded cost.

▶ **Lemma 3** (Raghvendra [28]). *For any i and $t > 1$, it holds that $c(M_i^*) \le t \cdot \mathrm{OPT}_i$, where $\mathrm{OPT}_i \coloneqq c(M_i^{\mathrm{OPT}})$ is denotes the cost of an optimal offline matching M_i^{OPT} of the first i requests.*

For constructing the online matching M_i', augmentation along a path is impossible since using recourse is not permitted. Instead, the online t-net-cost algorithm maintains M^* as an auxiliary matching and constructs M_i' by directly connecting the end points r_i and s_i of the augmenting path P_i. That is, $M_i' \coloneqq M_{i-1}' \cup \{(r_i, s_i)\}$. In particular, M_i' and M_i^* utilize the same sets of servers.

[1] For two sets X, Y, their symmetric difference is given by $X \oplus Y \coloneqq (X \cup Y) \setminus (X \cap Y)$. For matchings M_1, M_2, their symmetric difference $M_1 \oplus M_2$ consists of disjoint alternating paths and cycles.

Intuitively, in putting a higher weight on edges that would be added to M^* during augmentation, the parameter t in the t-net cost function discourages the offline t-net-cost algorithm from choosing long paths for augmentation (w.r.t. actual length). This allows for a trade-off between minimizing the cost of the underlying offline matching, and the connection costs in M' (with the latter in a greedy fashion). Looking at the extremal cases, this becomes even more clear. When $t = 1$, the offline t-net-cost algorithm is in fact equivalent to the Hungarian Method [22] which computes an optimal offline solution. The corresponding online matching, however, has a competitive ratio of $\Omega(n)$. In contrast, as t tends to infinity, the algorithm's behavior resembles that of the greedy online algorithm matching a request to the nearest free server. Its competitive ratio can be exponential [19]. Interestingly though, when $t = 3$, the t-net-cost algorithm has a competitive ratio of $O(\log n)$ [29].

3 A Constant-Competitive Algorithm

We start by giving an overview of our algorithm for the recourse model. It exploits the properties of the t-net-cost algorithm by carefully balancing between the offline matching M^* and the online matching M', simultaneously bounding competitive ratio and recourse budget. On a high level, this is done as follows. When a request arrives, we match it as in M' and locally group it with other recent requests into *blocks* that partition the line. Matching requests as in the online matching within a block somewhat increases total cost but requires zero recourse. A structural result, Lemma 8, allows us to bound this increase in cost up to a certain point at which the requests in the block are matched according to M^* causing a local update. During such an update, which we call a *recourse step*, the changes in M^* caused by the arrivals of the requests in the respective block are applied simultaneously, eliminating any redundant recourse actions. Intuitively, blocks can therefore be seen as input buffers for M^* that temporarily use edges from M'. The underlying structure of the blocks guarantees that recourse steps affect only the corresponding portion of the line. To prevent the recourse steps from causing too many reassignments, we additionally incorporate an edge freezing scheme that targets low-cost edges and at the same time keeps the overall cost low.

Given the precise value of OPT and n a priori, one could employ a very simple freezing scheme, which freezes all edges of M' with cost $\frac{\mathrm{OPT}}{n}$ or less. However, in the online model, we do not know OPT or n and, thus, need a dynamic freezing scheme. A typical guess-and-double approach may work concerning the costs. Yet, care has to be taken as $\frac{\mathrm{OPT}_i}{i}$ is not monotone. A major obstacle appears to be the bounding of the recourse budget. Details on our algorithm and dynamic freezing scheme are given in Section 3.

In Section 4, we consider alternating instances. Again, we simulate the offline matching M^* and employ a simple freezing scheme. After a request reaches a certain threshold of reassignments, we freeze this request and the currently associated matching edge. We charge detours that are taken due to frozen edges to large non-frozen edges of M^{OPT}.

3.1 Further Definitions and Notation

Our algorithm classifies requests according to the structure of intervals that describe where on the line the t-net-cost algorithm searches for a free server. Define the *search interval* of a request r_i as the open interval $\bar{\mathrm{I}}_i = (s_i^{\mathrm{L}}, s_i^{\mathrm{R}})$, where s_i^{L} and s_i^{R} are points on the line farthest to the left and right of r_i, respectively, reachable from r_i with t-net-cost $\phi_t^{M_{i-1}^*}(P_i)$. One of $s_i^{\mathrm{L}}, s_i^{\mathrm{R}}$ is the server s_i (which r_i is matched to in M'), see Lemma 6, while the other may not necessarily be a point of $R \cup S$. For the purposes of this definition, we think of it as a (virtual) server. That is, we ask the question "If point p was a server in S, would

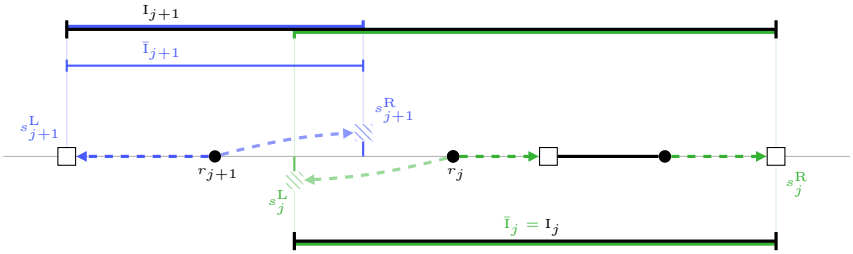


Figure 1 Construction of search intervals and aggregate search intervals of requests r_j and r_{j+1}. The points s^{L}_j and s^{R}_{j+1} (hatched) are points on the line that do not lie in S. We think of them as virtual servers for the purpose of defining $\bar{\mathrm{I}}_j$ and $\bar{\mathrm{I}}_{j+1}$ respectively.

we be able to reach it with t-net-cost $\phi_t^{M^*_{i-1}}(P_i)$?". In other words, $\bar{\mathrm{I}}_i$ is the convex hull of all points on the line, reachable from r_i via an augmenting path of t-net-cost (strictly) less than $\phi_t^{M^*_{i-1}}(P_i)$.

Define the *aggregate search interval* of r_i to be the maximal (open) interval I_i which contains r_i and is a subset of $\bigcup_{j\le i}\bar{\mathrm{I}}_j$. Intuitively, $\bigcup_{j\le i}\bar{\mathrm{I}}_j$ consists of the (disjoint) portions of the line, which the t-net-cost algorithm has considered, up to time i, in its search for free servers; the interval I_i is simply the portion containing r_i. See Figure 1 for an illustration. By definition, the portions of the line constituting $\bigcup_{j\le i}\bar{\mathrm{I}}_j$ grow monotonously (and possibly merge). Thus, the aggregate search intervals inhibit a laminar structure as detailed in the following observation.

▷ Observation 4. Whenever $i < j$, then either $\mathrm{I}_i \cap \mathrm{I}_j = \emptyset$ or $\mathrm{I}_i \subseteq \mathrm{I}_j$.

Another important observation is the fact, that the arrival of a request r_i only affects requests and servers in its search interval $\bar{\mathrm{I}}_i$. This is due to the fact that the augmenting path used by the t-net-cost algorithm is entirely contained in the search interval. Outside of $\bar{\mathrm{I}}_i$, the matchings M^* and M' remain unchanged motivating the following observation.

▷ Observation 5. Altering the arrival order of requests via a permutation π for which $\mathrm{I}_i \subseteq \mathrm{I}_j$ and $i < j$ imply $\pi(i) < \pi(j)$ does not alter the final matching.

We say an aggregate search interval I_i has *level* k, if $(1+\varepsilon)^{k-1} \le |\mathrm{I}_i| < (1+\varepsilon)^k$ and write $\ell(\mathrm{I}_i) = k$. Throughout, we set $t = 3$ and $\varepsilon = \frac{1}{32t}$. Further, two aggregate search intervals are said to belong to the same *block*, if they intersect with each other and are of same level. If the aggregate search intervals of a block do not intersect those of higher level, then this block is said to be a *top block*. With Observation 5 in mind, we note that the top blocks partition the line into portions that are compatible with the structure of M^*. A typical block-structure is depicted in Figure 2.

Our definition of search intervals was motivated by intuition and practicality (specifically for the proof of Lemma 12 later on). However, it describes intervals different from the search intervals defined in [29]. The following lemma shows that our definition of aggregate search intervals coincides with Raghvendra's definition of *intervals of a cumulative search region*. We may therefore use the corresponding results from [29].

▶ **Lemma 6.** *For a request r_i, the corresponding aggregate search interval I_i and interval of a cumulative search region C_i are equal. Further, for the search interval $\bar{\mathrm{I}}_i = (s^{\mathrm{L}}_i, s^{\mathrm{R}}_i)$, we have $s_i \in \{s^{\mathrm{L}}_i, s^{\mathrm{R}}_i\}$.*

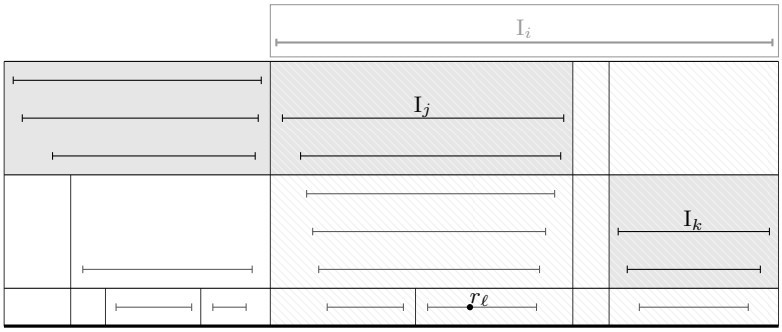


Figure 2 Illustration of a typical block structure. On arrival of r_i, its aggregate search interval I_i creates a top block (active) initiating a recourse step. Requests in the hatched area (now inactive) are reassigned w.r.t. $M^*_{i-1} \oplus M^{\mathrm{F}}_i$. On I_i, locally, $M^*_{i-1} = M^*_h$ for $j, k \leq h < i$.

Proof. To see that the first claim holds, note that the intervals C_i and I_i are constructed the same way. They are built by by taking the union of all known search intervals (for the respective definition) and choosing of the resulting new intervals that which contains the considered request. The definition of search intervals in [29], which, to avoid confusion, we call *dual intervals*, is as follows. The t-net-cost algorithm maintains dual values $y : S \cup R \to \mathbb{R}^+$ satisfying $y_s + y_r = c_{(s,r)}$ if $(s,r) \in M^*$ and $y_s + y_r \leq t \cdot c_{(s,r)}$ otherwise. Additionally, duals of free requests or servers are zero. When a request r_i arrives, a shortest t-net-cost path P_i is found and the duals of all vertices in the search tree (denoted by $A_i \subseteq S$ and $B_i \subseteq R$) are updated before augmentation so that the dual constraints on P_i are tight. This is true for both augmenting paths P^{L}_i and P^{R}_i that are used to reach s^{L}_i and s^{R}_i respectively. A dual interval of a request r_i is defined as $\mathit{interior}(\bigcup_{r \in B_i} \mathit{cspan}(r,i))$, where $\mathit{cspan}(r,i) = [r - \frac{y^i_{\max}(r)}{t}, r + \frac{y^i_{\max}(r)}{t}]$ and $y^i_{\max}(r)$ is the highest dual weight assigned to r until time i.

Raghvendra [28] shows, that the dual constraints on P^{L}_i and P^{R}_i are tight before augmentation. Therefore, we obtain our search intervals I_i by replacing $y^i_{\max}(r)$ with the dual weight of r before augmentation along P_i. Hence, a search interval is contained in the dual interval and therefore $\mathrm{I}_i \subseteq C_i$. At the same time, dual weights of requests are increased when the request is contained in some B_i as detailed above or reduced right after an augmentation. Thus, the maximal value $y^i_{\max}(r)$ is attained right before an augmentation, in which case the request is part of some P^{L}_h or P^{R}_h, implying $C_i \subseteq \mathrm{I}_i$. Therefore, the intervals I_i and C_i coincide and we may use the respective results from [29].

The second claim follows directly from Lemma 6 in [29], which states that there are no free servers in the dual interval. As seen above, search intervals are contained in dual intervals, so this statement is true for our definition as well. And lastly, since clearly $s_i \in [s^{\mathrm{L}}_i, s^{\mathrm{R}}_i]$, it follows that $s_i \in \{s^{\mathrm{L}}_i, s^{\mathrm{R}}_i\}$. ◀

3.2 The Algorithm

Our algorithm uses the above structures to partition the requests (dynamically) into three groups. For each group, we follow a different assignment procedure to construct our recourse matching M. We first give some intuition and then, further below, the precise description.

Group 1: The first group consists of all requests whose aggregate search intervals belong to a top block. We label requests in this group *active* and all other requests *inactive*. In M, we match an active request r_i exactly as in the online matching M'. That is, we have e_i

in M, where $e_i = (r_i, s_i) \in M'$ connects the endpoints of the minimum t-net-cost augmenting path P_i with respect to M^*_{i-1} (Algorithm 1, Step 1). In particular, any arriving request belongs immediately to a top block and is labeled active, by definition of aggregate search intervals. In the course of the algorithm's execution new blocks may appear (on top) rendering previously active requests inactive. We call this a *recourse step*.

Group 2: This group consists of inactive requests r_i, whose corresponding edge $e_i \in M'$ is of negligible size. The precise freezing scheme (Algorithm 1, Steps 2-4) is detailed below. Intuitively, it is not worth to spend recourse on them; it would ruin our recourse budget, see Figure 4. We call such a request or edge *frozen* and denote by $F_j \subseteq M'_j$ the set of frozen edges at time j. These requests, too, are to be matched in M according to the online matching M'. However, for technical reasons, the update on M for a newly frozen request is implemented with a subsequent recourse step. Denote by $M^{\mathrm{F}}_j \subseteq F_j$ the subset of frozen edges that is in $M_j \cap M'$. Their low costs ensures that frozen edges contribute in total at most OPT to the cost of M.

Group 3: The remaining requests (inactive and unfrozen) are in this group. Ideally, we would like to match these as in the offline matching M^*. This may not be possible, as M^* could assign a request r to a server s that is already covered by M^{F}. In such a case, we match r via a detour of low additional cost as follows; we call this *detour matching*.

Detour Matching. Consider an unfrozen request r and the symmetric difference $M^* \oplus M^{\mathrm{F}}$ consisting of alternating paths and cycles. There exists a (unique) alternating path P from r to some server s' not yet matched via M^{F}. To see this, note that the path starts with the edge $(r, s) \in M^*$ which cannot be contained in an alternating cycle as r is not frozen. Additionally, it cannot end at a request, since every request on P is reached via an edge of M^{F}. But since a request on P matched in M^{F} is certainly also matched (via a different edge) in M^*, it can be used to extend P. Thus, P ends at a server s' not matched in M^{F}. In M, we match r directly to this server s' (Alg. 1, Step 7). While the online matching M' does not change but only gets revealed over time, the offline matching M^* can change its structure drastically at any point. As this causes a lot of recourse, the first group is used as an input buffer and the detour matching is updated only periodically and locally, whenever a new top block appears. Specifically, at time i, an inactive request r_j is matched with respect to the detour matching $M^*_{h(i,j)} \oplus M^{\mathrm{F}}_i$, where $h(i,j) = \max\{k \leq i \mid r_j \in \mathrm{I}_k$ and I_k is not in a top block$\}$. That is, $h(i,j)$ is the last time before i when r_j participated in a recourse step. For an example, see Figure 2, where $h(i,\ell) = j$.

When considering active, inactive or frozen objects, or membership in a block, we identify a request r_i, the edge $e_i \in M'$ and the interval I_i. For instance, we say r_i is of level k and belongs to a certain block, when this holds for I_i, or, e_i is active when this is the case for r_i.

We describe Algorithm 1 in a step-by-step manner and detail the precise freezing scheme. On arrival of a request r_i, label it active and match it according to the online matching M'.

Freezing/unfreezing inactive requests. We update the set of frozen edges as follows. If the cost of an inactive edge $e_j \in M'$ falls below $\frac{\mathrm{OPT}_i}{i^2}$, then freeze it and add it to F_i. Note that while we use e_j to determine whether r_j is frozen, r_j may use another edge in M_i.

If for a previously frozen e_j, we have $c_{e_j} > \frac{\mathrm{OPT}_i}{i}$, unfreeze and delete it from F_i and M^{F}_i. Matching it according to M' is now too costly. Reassign it w.r.t. $M^*_{h(i,j)} \oplus M^{\mathrm{F}}_i$ as follows.

Consider an edge $e = (s, r)$ right before it is unfrozen. If e is already matched according to the offline matching, then there is nothing to do. Otherwise, it must be part of an alternating path or cycle in $M^* \oplus M^{\mathrm{F}}$. In the latter case, again, no recourse action needs to be taken as

Algorithm 1 for computing an online bipartite matching with recourse on the line.

On arrival of the i-th request r_i: ▷ r_i matched in M' via $e_i = (r_i, s_i)$
1: match r_i to s_i as in M' ▷ active
Freezing/Unfreezing
2: determine the set F_i of frozen edges
3: **for** inactive requests r_j that become unfrozen **do**
4: remove e_j from M_i^{F} and repair assignments on corresponding path of $M^*_{h(i,j)} \oplus M_i^{\mathrm{F}}$
Recourse step (new top block)
5: **if** there is no $j < i$ such that $\mathrm{I}_j \subseteq \mathrm{I}_i$ and $\ell(\mathrm{I}_j) = \ell(\mathrm{I}_i)$ **then**
6: **for** $r_j \in \mathrm{I}_i$ recently frozen **do** add $e_j = (r_j, s_j)$ to M_i^{F} and reassign r_j to s_j
7: **for** unfrozen $r_j \in \mathrm{I}_i \setminus \{r_i\}$ **do** reassign r_j according to $M^*_{i-1} \oplus M_i^{\mathrm{F}}$ ▷ now inactive



Figure 3 Illustration of Step 4 in Algorithm 1. Edges of M^* are snaked, edges of M^{F} solid and edges of $M \setminus M^{\mathrm{F}}$ dashed. After unfreezing r, the removal of (r, s) splits the path in $M^* \oplus M^{\mathrm{F}}$ in two parts.

the removal of e results in a path from r to s. In the detour matching, we want to connect the ends of this path, which is already accomplished by the edge $e = (s, r)$. Consider the case that $e \in P$ for some alternating path P in $M^* \oplus M^{\mathrm{F}}$ that starts in a request r' and ends in a server s'. Unfreezing e and matching according to $M^* \oplus M^{\mathrm{F}}$ decomposes P into the r'-s-path P_1 and the r-s'-path P_2. In Algorithm 1, we implement these changes via two recourse actions: we reassign r to s' and r' to s; see Figure 3.

Recourse step. If there is no $j < i$ such that $\mathrm{I}_j \subseteq \mathrm{I}_i$ and $\ell(\mathrm{I}_j) = \ell(\mathrm{I}_i)$, then the arrival of r_i produces a new top block and may render a number of previously active requests inactive. This triggers, what we call a *recourse step* involving a number of recourse actions to accommodate the newly inactive requests as follows.

First, assign requests in I_i that were recently frozen according to M' and add the corresponding edge to M_i^{F}. Next, reassign all other requests (non-frozen, inactive) that lie in I_i according to $M^*_{i-1} \oplus M_i^{\mathrm{F}}$. We described this as detour matching above; see Figure 2.

Delaying the reassignment of frozen requests according to the frozen edge until their next recourse step, prevents reassignments when a request repeatedly alternates between frozen and unfrozen. This "delayed reassignment" accounts for the difference between M_i^{F} and F_i.

To see that the freezing scheme is indeed necessary, consider the example in Figure 4 showing an alternating instance with exponentially increasing edge costs. Without freezing low-cost edges, at the arrival of any request r_i with $i > 2$, all requests r_j with $j < i$ are reassigned. Thus, the recourse budget is linear.

Bounding the competitive ratio. The work already put into structuring M enables us to bound its cost rather easily. Recall that $c(M^*) \leq t \cdot \mathrm{OPT}$ due to Lemma 3. Frozen edges satisfy $c(M^{\mathrm{F}}) \leq \mathrm{OPT}$ as, after arrival of the last request, there are at most n frozen edges of cost at most $\frac{\mathrm{OPT}}{n}$ each. By triangle inequality, the contribution of inactive non-frozen requests to the cost of M is at most $c(M^* \oplus M^{\mathrm{F}}) \leq (t+1) \cdot \mathrm{OPT}$.

▷ Observation 7. The cost of inactive edges is at most $(t+2) \cdot \mathrm{OPT}$.

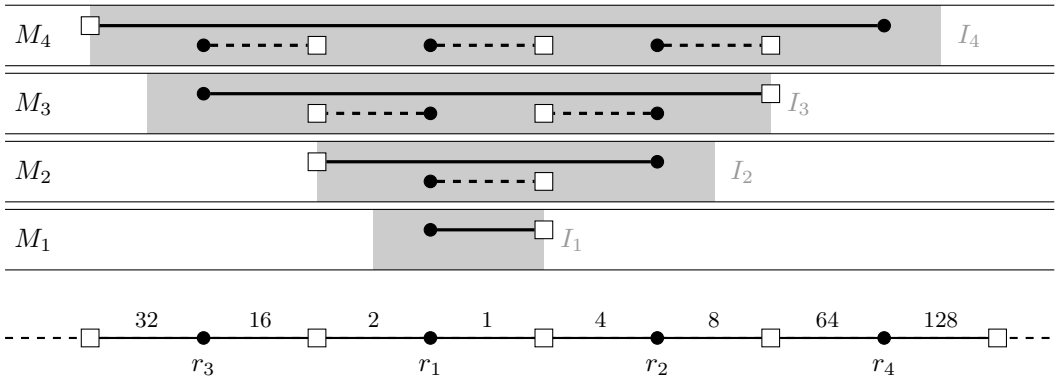


Figure 4 An alternating instance with exponentially increasing connection costs. Edge costs are indicated above the corresponding portion of the line (drawing not to scale). Note that the aggregate search intervals grow exponentially in size. Therefore, a (top) block consists of a single aggregate search interval only and every arrival of a request triggers a recourse step. Active edges are drawn solid, inactive edges dashed.

To bound the cost of active edges, we build on the analysis of Raghvendra [29]. We refine his technical propositions and perform a slightly more fine-grained analysis. Instead of simultaneously bounding the cost of all blocks of the same level, we argue more generally on the cost of any set of disjoint blocks at different levels. In particular, we are interested in bounding the cost contribution of the top blocks.

▶ **Lemma 8.** *For $t = 3$, the cost of all active edges of M is bounded by $O(\mathrm{OPT})$.*

To prove this lemma, we may closely follow the argumentation given in [29] for the online problem without recourse. In fact, there is given essentially the same statement for all edges of blocks of a particular level. We can adapt the approach, crucially using Observation 5, to make the arguments work for (disjoint) blocks of different levels. We leave the detailed proof for the full version.

▶ **Corollary 9.** *Algorithm 1 has a constant competitive ratio.*

Bounding the recourse. Due to delayed reassignments, freezing a request r does not cause any immediate recourse actions. Unfreezing r, on the other hand, may cause reassignments (Step 4), namely to r and possibly one additional request, see Figure 3. We charge these two recourse actions to r, in particular to the last recourse step it participated in. However, recourse due to unfreezing r happens at most once between two consecutive recourse steps of r. This implies, that a request is charged at most three times per recourse step it participates in, once for the recourse step itself and possible two times more for a subsequent unfreezing.

▷ Observation 10. A request is charged at most three times per recourse step it is involved in.

For a request getting frozen at time i and unfrozen at time $j > i$, we have $c_e \leq \frac{\mathrm{OPT}_i}{i^2}$ and $c_e > \frac{\mathrm{OPT}_j}{j}$ for the corresponding $e \in M'$. As $\mathrm{OPT}_i \leq \mathrm{OPT}_j$, this implies the following.

▷ Observation 11. A request frozen at time i stays frozen until time at least i^2.

On the other hand, the number of recourse steps in which a continuously non-frozen request takes part in can be bounded from above.

▶ **Lemma 12.** *If a request $r \in \mathrm{I}_i \subseteq \mathrm{I}_j$ is not frozen from time i to time j, then it holds that $\ell(\mathrm{I}_j) - \ell(\mathrm{I}_i) = O(\log j)$. In particular, between time i and j, there are $O(\log j)$ recourse steps in which r can participate.*

Proof. Consider the search interval $\bar{\mathrm{I}}_j = (s_j^{\mathrm{L}}, s_j^{\mathrm{R}})$. By definition of search intervals, there exist augmenting paths $P_j^{\mathrm{L}}, P_j^{\mathrm{R}}$ connecting r_j to s_j^{L} and s_j^{R} with the same t-net-cost as P_j, the path used by the t-net-cost algorithm. We bound the length of $P_j \in \{P_j^{\mathrm{L}}, P_j^{\mathrm{R}}\}$ by

$$c(P_j) = c(P_j \cap M_{j-1}^*) + c(P_j \cap M_j^*) \leq 2t \cdot \mathrm{OPT}_j. \tag{2}$$

Without loss of generality, assume $P_j = P_j^{\mathrm{L}}$. Interpreting the point s_j^{R} as a virtual server and assuming that it is a contained in S, we could augment M_{j-1}^* also along P_j^{R} yielding a different matching $\tilde{M}_j^*$. By definition of P_j^{R} and Equation (1), we have

$$t \cdot c(P_j \cap M_j^*) - c(P_j \cap M_{j-1}^*) = \phi_t^{M_{j-1}^*}(P_j) = \phi_t^{M_{j-1}^*}(P_j^{\mathrm{R}}) = t \cdot c(P_j^{\mathrm{R}} \cap \tilde{M}_j^*) - c(P_j^{\mathrm{R}} \cap M_{j-1}^*).$$

Thus,

$$\begin{aligned} c(P_j^{\mathrm{R}}) &= c(P_j^{\mathrm{R}} \cap M_{j-1}^*) + c(P_j^{\mathrm{R}} \cap \tilde{M}_j^*) \\ &\leq c(P_j^{\mathrm{R}} \cap M_{j-1}^*) + c(P_j \cap M_j^*) + \tfrac{1}{t} c(P_j^{\mathrm{R}} \cap M_{j-1}^*) \leq 3t \cdot \mathrm{OPT}_j. \end{aligned}$$

From Equation (2), we get $|\bar{\mathrm{I}}_j| \leq c(P_j^{\mathrm{L}}) + c(P_j^{\mathrm{R}}) \leq 5t \cdot \mathrm{OPT}_j$, which implies $|\mathrm{I}_j| \leq \sum_{k \leq j} |\bar{\mathrm{I}}_k| \leq 5t \cdot j \cdot \mathrm{OPT}_j$. On the other hand, the cost of $|\mathrm{I}_i|$ can be bounded from below by c_e, where e is the edge in M' incident to r, as both ends of e are contained in the interval. We then obtain

$$\frac{|\mathrm{I}_j|}{|\mathrm{I}_i|} \leq \frac{5t \cdot j \cdot \mathrm{OPT}_j}{c_e} < 5t \cdot j^3.$$

The last inequality follows from the assumption that r is not frozen at time j, implying that $c_e > \frac{\mathrm{OPT}_j}{j^2}$. Recall that $|\mathrm{I}_i| < (1+\varepsilon)^{\ell(\mathrm{I}_i)}$ and $(1+\varepsilon)^{\ell(\mathrm{I}_j)-1} \leq |\mathrm{I}_j|$. We conclude

$$\ell(\mathrm{I}_j) - \ell(\mathrm{I}_i) \leq 1 + \log_{(1+\varepsilon)}(5t \cdot j^3) = 1 + \log_{(1+\varepsilon)}(5t) + \frac{\log j}{\log(1+\varepsilon)} \leq \; c \cdot \log j\,, \tag{3}$$

for some constant c. Regarding the second claim, recall that r participates in a recourse step when an interval containing r opens a new top block (i.e. a new level) while r is not (freshly) frozen. Between time i and time j, this can only happen for intervals I_h of distinct levels for which $\mathrm{I}_i \subseteq \mathrm{I}_h \subseteq \mathrm{I}_j$. The claim follows. ◀

Lemma 12 and Observation 11 bound the total number of recourse actions taken by Algorithm 1.

▶ **Lemma 13.** *Algorithm 1 uses a recourse budget of at most $O(\log n)$.*

Proof. Consider a request r. By Observation 10, it suffices to bound the number of recourse steps that r is involved in. Let $[i_h^U, i_h^F]$, for $h = 0, 1, \ldots, k$, be maximal intervals of consecutive time points during which r is not frozen, i.e., r is not frozen at any time $i \in [i_h^U, i_h^F]$. We use induction on k to show that r participates in at most $2c \cdot \log(i_k^F)$ recourse steps, where c is the constant from Equation (3). The base case, $k = 0$, follows directly from Lemma 12. For $k \geq 1$, we have $(i_{k-1}^F)^2 \leq i_k^U \leq i_k^F$ due to Observation 11. By induction hypothesis, the number of reassignments that involve r in the first $k-1$ time intervals is at most

$$2c \cdot \log(i_{k-1}^F) \leq 2c \cdot \log\left(\sqrt{i_k^F}\right) = c \cdot \log(i_k^F).$$

For the last time interval, we have at most $c \cdot \log(i_k^F)$ many such recourse steps by Lemma 12. Since $i_k^F \leq n$, this concludes the proof. ◀

Corollary 9 and Lemma 13 together imply Theorem 1.

4 A Near-Optimal Algorithm on Alternating Instances

For alternating instances, we may assume that requests and servers alternate from $-\infty$ to ∞ on the line, with servers at $\pm\infty$. For such instances, an optimum matching matches all requests either to the server directly to their left or all requests to the server on their right. Denote these matchings by M^{L} and M^{R} respectively and call their edges *minimal*.

We describe a $(1+\varepsilon)$-competitive algorithm for alternating instances that reassigns each request at most a constant number of times. In addition to its output M, it maintain M^* and a set of frozen edges M^{F}. A request is frozen when it is reassigned the k-th time, for some k only depending on ε. The request remains matched to its current server perpetually and the corresponding edge is added to M^{F}. Non-frozen requests are matched according to the detour matching $M^* \oplus M^{\mathrm{F}}$ as described in Section 3. By design, the recourse budget per request is constant, only the competitive analysis remains.

Notation. We use a similar interval structure as before and keep the notation. Consider intervals $\mathrm{I}_i = [s_i^{\mathrm{L}}, s_i^{\mathrm{R}}]$, where $s_i^{\mathrm{L}}, s_i^{\mathrm{R}} \in S$ are the closest free servers on the line to the left and right of r_i respectively at the time of its arrival. Denote by $P_i^{\mathrm{L}}, P_i^{\mathrm{R}}$ the alternating paths connecting r to s_i^{L} and s_i^{R} respectively that have shortest t-net-cost.

Considered as line segments, the augmenting paths P_i have a laminar structure. We view them as nodes of a forest, where P_i is a child of the minimal augmenting path that properly contains it, or a root if no such path exists (see Figure 5). The *depth* of a path P_i denotes its distance to the root and determines the number of reassignments of the corresponding request r_i in M^*.

▶ **Lemma 14.**

(i) *The paths $P_i^{\mathrm{L}}, P_i^{\mathrm{R}}$ and the matching M_i^* only use minimal edges.*

(ii) *If $P_i = P_i^X$, for $X \in \{\mathrm{L}, \mathrm{R}\}$, then in the area of I_i, locally, we have that $M_i^* = M^{\mathrm{X}}$. Specifically, this implies $\phi_t^{M_{i-1}^*}(P_i) = \phi_t^{M^Y}(P_i)$ for $X \neq Y \in \{\mathrm{L}, \mathrm{R}\}$.*

(iii) *If P_j is a child of P_i, then $P_i = P_i^{\mathrm{L}}$ if and only if $P_j = P_j^{\mathrm{R}}$. In particular, we have $\mathrm{I}_j \subseteq P_i$ and $\mathrm{I}_j \cap M_j^* \cap M_i^* = \emptyset$.*

Proof. (i): We use induction on i. The base case, $i = 1$, is easy. Let $i \geq 2$. Without loss of generality $P_i = P_i^{\mathrm{R}}$. Assume P_i contains a non-minimal edge e. Let the notation (s, r) for an edge reflect that s is to the left of r on the line. Edges $(s, r) \in P_i$ are contained in M_{i-1}^* and by induction hypothesis minimal, so $e = (r, s) \in M_i^*$. Consider the server s' right of r. Since e is not minimal, $r < s' < r' < s$, with r' being the server right of s'. If s' was free, altering P_i to go from r directly to s' would yield a lower cost. This follows from the fact that the t-net-cost of a path from a request to a free server is always non-negative, see [28]. If s' is matched, it must be matched to r'. Therefore, replacing e by $(r, s'), (s', r'), (r', s)$ reduces the cost as well, contradicting the minimality of P_i.

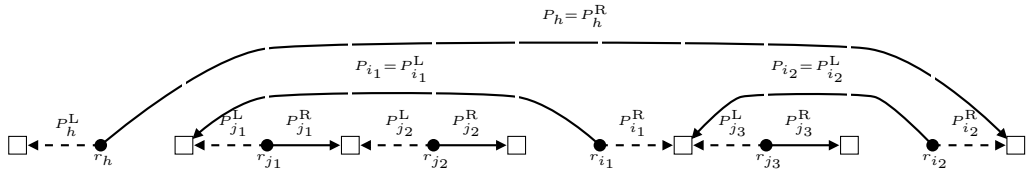


Figure 5 Illustration of a path tree in an alternating instance. Paths not chosen for augmentation are dashed. Servers are depicted as squares and requests as filled circles.

(ii): By statement (i), P_i^{L} and P_i^{R} only consist of minimal edges. Then $P_i^{\mathrm{L}} \cap M_{i-1}^* = M^{\mathrm{R}}$ and $M_{i-1}^* \cap P_i^{\mathrm{R}} = M^{\mathrm{L}}$. After augmenting M_{i-1}^* along P_i, the matching is flipped (locally).

(iii): By (ii), we know that $M_j^* \cap \mathrm{I}_j = M^{\mathrm{X}}$, say $X = R$, so edges are of the form (r, s). From part (i), only augmenting paths $P_h = P_h^{\mathrm{L}}$ can traverse them. If this happens, all of I_j is traversed as there is no free server in its interior. As parent of P_j, P_i is the first path to properly contain P_j and thus $P_i = P_i^{\mathrm{L}}$ and in particular $\mathrm{I}_j \subseteq P_i$. Therefore, the equation $\mathrm{I}_j \cap M_j^* \cap M_i^* = \emptyset$ follows directly from (ii). ◀

The above lemma can be used to show that the sum of lengths of augmenting paths of some depth grows exponentially towards the root. For a path P_h, denote by $|P_h|$ the length of the corresponding line segment. Further, let $\mathcal{H}_k$ be the set of indices of paths at depth k in the induced subtree of P_h with the root P_h at depth 0.

▶ **Lemma 15.** *Consider a path P_h and its grandchildren P_j, for $j \in \mathcal{H}_2$. Then*

$$|P_h| \geq \left(2 - \tfrac{1}{t}\right) \cdot \sum_{j \in \mathcal{H}_2} |P_j|.$$

Proof. Denote by P_i, $i \in \mathcal{I} = \mathcal{H}_1$, children of P_h in the path-forest and by $\mathcal{J}_i \subseteq \mathcal{J} = \mathcal{H}_2$ the sets of indices of their respective children. Without loss of generality, assume $P_h = P_h^{\mathrm{R}}$. Lemma 14, (iii), implies $P_i = P_i^{\mathrm{L}}$, and $P_j = P_j^{\mathrm{R}}$, for $i \in \mathcal{I}, j \in \mathcal{J}$; see Figure 5. With again Lemma 14, (iii), and $\phi_t^{M_{i-1}^*}(P_i^{\mathrm{R}}) \geq \phi_t^{M_{i-1}^*}(P_i^{\mathrm{L}})$, we get

$$\begin{aligned} t \cdot |P_h| &\geq t \cdot \sum_{i \in \mathcal{I}} |I_i| = t \cdot \sum_{i \in \mathcal{I}} \left(|P_i^{\mathrm{L}}| + |P_i^{\mathrm{R}}|\right) \geq \sum_{i \in \mathcal{I}} \left(t \cdot |P_i^{\mathrm{L}}| + \phi_t^{M_{i-1}^*}(P_i^{\mathrm{R}})\right) \\ &\geq t \cdot \sum_{j \in \mathcal{J}} |P_j^{\mathrm{R}}| + t \cdot \sum_{i \in \mathcal{I}} |P_i \setminus (\cup_{j \in \mathcal{J}_i} P_j^{\mathrm{R}})| + \sum_{i \in \mathcal{I}} \phi_t^{M_{i-1}^*}(P_i^{\mathrm{L}}). \end{aligned} \tag{4}$$

Using Lemma 14, we get

$$\begin{aligned} \phi_t^{M_{i-1}^*}(P_i^{\mathrm{L}}) = \phi_t^{M^{\mathrm{R}}}(P_i^{\mathrm{L}}) &= \sum_{j \in \mathcal{J}_i} \left(\phi_t^{M^{\mathrm{R}}}(P_j^{\mathrm{L}}) + \phi_t^{M^{\mathrm{R}}}(P_j^{\mathrm{R}})\right) + \phi_t^{M^{\mathrm{R}}}(P_i \setminus (\cup_{j \in \mathcal{J}_i} \mathrm{I}_j)) \\ &\geq \sum_{j \in \mathcal{J}_i} \left(\phi_t^{M_{j-1}^*}(P_j^{\mathrm{L}}) + \phi_t^{M^{\mathrm{R}}}(P_j^{\mathrm{R}})\right) - t \cdot |P_i \setminus (\cup_{j \in \mathcal{J}_i} \mathrm{I}_j)|. \end{aligned}$$

Since $\phi_t^{M_{j-1}^*}(P_j^{\mathrm{L}}) \geq \phi_t^{M_{j-1}^*}(P_j^{\mathrm{R}}) = \phi_t^{M^{\mathrm{L}}}(P_j^{\mathrm{L}})$, the above together with Equation (4) implies

$$t \cdot |P_h| \geq \sum_{j \in \mathcal{J}} \left(t \cdot |P_j^{\mathrm{R}}| + \phi_t^{M^{\mathrm{L}}}(P_j^{\mathrm{L}}) + \phi_t^{M^{\mathrm{R}}}(P_j^{\mathrm{R}})\right) \geq (2t - 1) \cdot \sum_{j \in \mathcal{J}} |P_j^{\mathrm{R}}|,$$

where last inequality follows from the observation that $\phi_t^{M^{\mathrm{L}}}(P) + \phi_t^{M^{\mathrm{R}}}(P) = (t-1) \cdot |P|$. ◀

Proof of Theorem 2. We in fact prove a stronger result than in the theorem statement and show that the algorithm described in the beginning of this section is $(1+\varepsilon)$-competitive while reassigning each request at most $O(\varepsilon^{-(1+\lambda)})$ times for fixed $\lambda > 0$. Consider a path P_h. The intervals I_j, $j \in \mathcal{H}_{2k+2}$, are contained in paths $P_{j'}$, $j' \in \mathcal{H}_{2k+1}$, by Lemma 14, (iii). Lemma 15 implies

$$\textstyle\sum_{j \in \mathcal{H}_{2k+2}} |I_j| \leq \sum_{j' \in \mathcal{H}_{2k+1}} |P_{j'}| \leq \left(2 - \frac{1}{t}\right)^{-k} \cdot \sum_{i \in \mathcal{H}_1} |P_i|. \tag{5}$$

Raghvendra [28] shows that the t-net-cost of augmenting paths is always non-negative. In particular, $\phi_t^{M_{h-1}^*}(P_h) = t \cdot c(P_h \cap M_h^*) - c(P_h \cap M_{h-1}^*) \geq 0$, and thus

$$|P_h| = c(P_h \cap M_h^*) + c(P_h \cap M_{h-1}^*) \leq (t+1) \cdot c(P_h \cap M_h^*). \tag{6}$$

Similarly, $\sum_{i\in\mathcal{H}_1}|P_i| \le (t+1)\cdot\sum_{i\in\mathcal{H}_1} c(P_i\cap M_i^*) \le (t+1)\cdot c(P_h\cap M_i^*)$. In an interval I_i, locally, $M_h^* = M^{\mathrm{L}}$ if and only if $M_i^* = M^{\mathrm{R}}$ by Lemma 14 (ii). With (5) and (6), this implies

$$\left[\tfrac{1}{t+1}\left(2-\tfrac{1}{t}\right)^k - 1\right]\cdot \sum_{j\in\mathcal{H}_{2k+2}} |\mathrm{I}_j| \le c\left(P_h \setminus (\cup_{j\in\mathcal{H}_{2k+2}}\mathrm{I}_j)\cap M^{\mathrm{X}}\right), \quad X\in\{\mathrm{L},\mathrm{R}\}. \tag{7}$$

Denote by $\alpha(k,t)$ the term in square brackets. When a (minimal) edge is frozen, the remaining instance is again alternating and at most one request will take a detour due to (r,s) being frozen. The additional cost is bounded by $|\mathrm{I}_r|$ and can, via Equation (7), be charged to non-frozen parts of M^{OPT}. A part $P_h\cap M^{\mathrm{OPT}}$ is charged this way at most $2k+2$ times before P_h itself is frozen which leads to a competitive factor of $(t+\frac{2k+2}{\alpha(k,t)})$. Substituting $\alpha(k,t)$ from Equation (7) and setting $t = 1+\frac{\varepsilon}{2}$ and $k = \frac{4c(\lambda)}{\varepsilon}\cdot\frac{\varepsilon^{-\lambda}}{\lambda}$, for a constant $c(\lambda)$, we can show that this term is at most $1+\varepsilon$, yielding a recourse of $O_\lambda(\varepsilon^{-(1+\lambda)})$. ◀

As a byproduct, we show, for this special class of instances, a result in the online setting without recourse. It relates the competitive ratio to the cost metric, i.e., the maximum difference in edge cost for connecting a request to a server. This result compares to the best known competitive ratio of $O(\log n)$ by Raghvendra [29].

▶ **Theorem 16.** *The online t-net-cost algorithm is $O(\log\Delta)$-competitive for online matching on an alternating line, where $\Delta = \max_{r,r'\in R,s,s'\in S}\frac{c(r,s)}{c(r',s')}$.*

Proof. Consider an edge e_i of M' and assume $P_i = P_i^{\mathrm{R}}$. By Lemma 14, (ii),

$$t\cdot c(P_i^{\mathrm{L}}\cap M^{\mathrm{L}}) - c(P_i^{\mathrm{L}}\cap M^{\mathrm{R}}) = \phi_t^{M_{i-1}^*}(P_i^{\mathrm{L}}) \ge \phi_t^{M_{i-1}^*}(P_i) = t\cdot c(P_i\cap M^{\mathrm{R}}) - c(P_i\cap M^{\mathrm{L}}).$$

Therefore,

$$|P_i| = c(P_i\cap M^{\mathrm{L}}) + c(P_i\cap M^{\mathrm{R}}) \le \left(1+\tfrac{1}{t}\right)\cdot c(P_i\cap M^{\mathrm{L}}) + c(P_i^{\mathrm{L}}\cap M^{\mathrm{L}}).$$

As in Equation (6), we obtain $|P_i| \le (t+1)\cdot c(P_i\cap M^{\mathrm{R}})$. Together with the above, this implies that $c_{e_i} = |P_i| \le c(\mathrm{I}_i\cap M^{\mathrm{OPT}})\cdot\max\{t+1, 1+\frac{1}{t}\}$. Since intervals corresponding to the same depth in the path-forest are disjoint, we can bound the lengths of paths which are of same depth by $\max\{t+1,1+\frac{1}{t}\}\cdot\mathrm{OPT}$. As there are at most $2\cdot\log_{(2-1/t)}\Delta$ levels in total, by Lemma 15, the theorem's statement follows. ◀

5 Conclusion

In this paper, we give non-trivial results for the min-cost online bipartite matching problem with recourse. The results were obtained simultaneously with and independently of Gupta et al. [17] who consider also more general metrics than the line. We confirm that an average recourse of $O(\log n)$ per request is sufficient to obtain an $O(1)$-competitive matching on the line. It remains open if such a result can be obtained in a non-amortized setting, where the recourse is available only per iteration. Our algorithm is clearly designed for the amortized setting as it buffers online matching decisions and repairs them in batches.

Further, it remains open whether constant recourse per request is sufficient for maintaining an $O(1)$-competitive matching on the line, as for the case of alternating requests. This may be very well possible as there is, currently, no lower bound that rules this out.

Finally, we remind of a major open question in this field: Does there exist an $O(1)$-competitive algorithm for online matching on the line without any recourse?

References

1 S. Angelopoulos, C. Dürr, and S. Jin. Online maximum matching with recourse. In *MFCS*, volume 117 of *LIPIcs*, pages 8:1–8:15, 2018.

2 A. Antoniadis, N. Barcelo, M. Nugent, K. Pruhs, and M. Scquizzato. A $o(n)$-competitive deterministic algorithm for online matching on a line. *Algorithmica*, 81(7):2917–2933, 2019.

3 A. Antoniadis, C. Fischer, and A. Tönnis. A collection of lower bounds for online matching on the line. In *LATIN*, volume 10807 of *LNCS*, pages 52–65, 2018.

4 T. Avitabile, C. Mathieu, and L. Parkinson. Online constrained optimization with recourse. *Inf. Process. Lett.*, 113(3):81–86, 2013.

5 N. Bansal, N. Buchbinder, A. Gupta, and J. Naor. A randomized $o(\log^2 k)$-competitive algorithm for metric bipartite matching. *Algorithmica*, 68(2):390–403, 2014.

6 A. Bernstein, J. Holm, and E. Rotenberg. Online bipartite matching with amortized $o(\log^2 n)$ replacements. *J. ACM*, 66(5):37:1–37:23, 2019.

7 B. Bosek, D. Leniowski, P. Sankowski, and A. Zych. Online bipartite matching in offline time. In *FOCS*, pages 384–393, 2014.

8 K. Chaudhuri, C. Daskalakis, R. Kleinberg, and H. Lin. Online bipartite perfect matching with augmentations. In *INFOCOM*, pages 1044–1052, 2009.

9 R. Duan and S. Pettie. Linear-time approximation for maximum weight matching. *J. ACM*, 61(1):1:1–1:23, 2014.

10 B. Fuchs, W. Hochstättler, and W. Kern. Online matching on a line. *Theor. Comput. Sci.*, 332(1-3):251–264, 2005.

11 M. Gairing and M. Klimm. Greedy metric minimum online matchings with random arrivals. *Oper. Res. Lett.*, 47(2):88–91, 2019.

12 E. Grove, M.-Y. Kao, P. Krishnan, and J. Vitter. Online perfect matching and mobile computing. In *WADS*, volume 955 of *LNCS*, pages 194–205, 1995.

13 A. Gu, A. Gupta, and A. Kumar. The power of deferral: Maintaining a constant-competitive steiner tree online. *SIAM J. Comput.*, 45(1):1–28, 2016.

14 A. Gupta, G. Guruganesh, B. Peng, and D. Wajc. Stochastic online metric matching. In *ICALP*, volume 132 of *LIPIcs*, pages 67:1–67:14, 2019.

15 A. Gupta, A. Kumar, and C. Stein. Maintaining assignments online: Matching, scheduling, and flows. In *SODA*, pages 468–479, 2014.

16 A. Gupta and K. Lewi. The online metric matching problem for doubling metrics. In *ICALP*, volume 7391 of *LNCS*, pages 424–435, 2012.

17 V. Gupta, R. Krishnaswamy, and S. Sandeep. Permutation strikes back: The power of recourse in online metric matching. Accepted to appear in APPROX-RANDOM, 2020.

18 M. Imase and B. Waxman. Dynamic steiner tree problem. *SIAM J. Discrete Math.*, 4(3):369–384, 1991.

19 B. Kalyanasundaram and K. Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993.

20 R. Karp, U. Vazirani, and V. Vazirani. An optimal algorithm for on-line bipartite matching. In *STOC*, pages 352–358, 1990.

21 S. Khuller, S. Mitchell, and V. Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theor. Comput. Sci.*, 127(2):255–267, 1994.

22 H. W. Kuhn and B. Yaw. The hungarian method for the assignment problem. *Naval Res. Logist. Quart*, pages 83–97, 1955.

23 J. Lacki, J. Oćwieja, M. Pilipczuk, P. Sankowski, and A. Zych. The power of dynamic distance oracles: Efficient dynamic algorithms for the steiner tree. In *STOC*, pages 11–20, 2015.

24 J. Matuschke, U. Schmidt-Kraepelin, and J. Verschae. Maintaining perfect matchings at low cost. In *ICALP*, volume 132 of *LIPIcs*, pages 82:1–82:14, 2019.

25 N. Megow, M. Skutella, J. Verschae, and A. Wiese. The power of recourse for online MST and TSP. *SIAM J. Comput.*, 45(3):859–880, 2016.

26 A. Mehta. Online matching and ad allocation. *Found. Trends Theor. Comput. Sci.*, 8(4):265–368, 2013.

27 K. Nayyar and S. Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *FOCS*, pages 505–515, 2017.

28 S. Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *APPROX-RANDOM*, volume 60 of *LIPIcs*, pages 18:1–18:16, 2016.

29 S. Raghvendra. Optimal analysis of an online algorithm for the bipartite matching problem on a line. In *SoCG*, volume 99 of *LIPIcs*, pages 67:1–67:14, 2018.

30 Yongho Shin, Kangsan Kim, Seungmin Lee, and Hyung-Chan An. Online graph matching problems with a worst-case reassignment budget. *CoRR*, abs/2003.05175, 2020. `arXiv:2003.05175`.

Hardness of Approximation of (Multi-)LCS over Small Alphabet

Amey Bhangale
University of California Riverside, CA, USA
ameyrbh@gmail.com

Diptarka Chakraborty
National University of Singapore, Singapore
diptarka@comp.nus.edu.sg

Rajendra Kumar
IIT Kanpur, India
National University of Singapore, Singapore
rjndr2503@gmail.com

Abstract

The problem of finding longest common subsequence (LCS) is one of the fundamental problems in computer science, which finds application in fields such as computational biology, text processing, information retrieval, data compression etc. It is well known that (decision version of) the problem of finding the length of a LCS of an arbitrary number of input sequences (which we refer to as Multi-LCS problem) is NP-complete. Jiang and Li [SICOMP'95] showed that if Max-Clique is hard to approximate within a factor of s then Multi-LCS is also hard to approximate within a factor of $\Theta(s)$. By the NP-hardness of the problem of approximating Max-Clique by Zuckerman [ToC'07], for any constant $\delta > 0$, the length of a LCS of arbitrary number of input sequences of length n each, cannot be approximated within an $n^{1-\delta}$-factor in polynomial time unless P=NP. However, the reduction of Jiang and Li assumes the alphabet size to be $\Omega(n)$. So far no hardness result is known for the problem of approximating Multi-LCS over sub-linear sized alphabet. On the other hand, it is easy to get $1/|\Sigma|$-factor approximation for strings of alphabet Σ.

In this paper, we make a significant progress towards proving hardness of approximation over small alphabet by showing a polynomial-time reduction from the well-studied *densest k-subgraph* problem with *perfect completeness* to approximating Multi-LCS over alphabet of size $poly(n/k)$. As a consequence, from the known hardness result of densest k-subgraph problem (e.g. [Manurangsi, STOC'17]) we get that no polynomial-time algorithm can give an $n^{-o(1)}$-factor approximation of Multi-LCS over an alphabet of size $n^{o(1)}$, unless the Exponential Time Hypothesis is false.

2012 ACM Subject Classification Theory of computation → Problems, reductions and completeness

Keywords and phrases Longest common subsequence, Hardness of approximation, ETH-hardness, Densest k-subgraph problem

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.38

Category APPROX

Related Version A full version of the paper is available at https://arxiv.org/abs/2006.13449.

Funding *Diptarka Chakraborty*: Supported in part by NUS ODPRT Grant, WBS No. R-252-000-A94-133.
Rajendra Kumar: Supported in part by National Research Foundation Singapore under its AI Singapore Programme [Award Number: AISG-RP-2018-005].

Acknowledgements Authors would like to thank anonymous reviewers for providing helpful comments on an earlier version of this paper and especially for pointing out a small technical mistake in the proof of Lemma 16. Authors would also like to thank Pasin Manurangsi for pointing out that for certain regimes no hardness result is known for the densest k-subgraph problem.



© Amey Bhangale, Diptarka Chakraborty, and Rajendra Kumar;
licensed under Creative Commons License CC-BY



1 Introduction

Finding *longest common subsequence* (LCS) of a given set of strings over some alphabet is one of the fundamental problems of computer science. The computational problem of finding (the length of a) LCS has been intensively studied for the last five decades (see [16] and the references therein). This problem finds many applications in the fields of computational biology, data compression, pattern recognition, text processing and others. LCS is often considered among two strings, and in that case it is considered to be one of the classic string similarity measures (see [5]). The general case, when the number of input strings is unrestricted, is also very interesting and well-studied. To avoid any confusion we refer to this general version of the LCS problem as *Multi-LCS* problem. One of the major applications of Multi-LCS is to find similar regions of a set of DNA sequences. Multi-LCS is also a special case of the multiple sequence alignment and consensus subsequence discovery problem (e.g. [27]). Interested readers may refer to the chapter entitled "Multi String Comparison-the Holy Grail" of the book [13] for a comprehensive study on this topic. Other applications of Multi-LCS include text processing, syntactic pattern recognition [22] etc.

Using a basic dynamic programming algorithm [30] we can find a LCS between two strings of length n in quadratic time. However the general version, i.e., the Multi-LCS problem is known to be NP-hard [23] even for the binary alphabet. This problem remains NP-hard even with certain restrictions on input strings (e.g. [7]). For m input strings a generalization of the basic dynamic programming algorithm finds LCS in time $O(mn^m)$. Recently, Abboud, Backurs and Williams [2] showed that an $O(n^{m-\varepsilon})$ time (for any $\varepsilon > 0$) algorithm for this problem would refute the Strong Exponential Time Hypothesis (SETH) even for alphabet of size $O(m)$.

Due to the computational hardness of exact computation of a LCS, an interesting problem is what is the best approximation factor that we can achieve within a reasonable time bound. A c-approximate solution (for some $0 < c \leq 1$) of a LCS is a common subsequence of length at least $c \cdot |LCS|$, where $|LCS|$ denotes the length of a LCS. For the Multi-LCS problem, Jiang and Li [18] showed that if Max-Clique is hard to approximate within a factor of s then Multi-LCS is also hard to approximate within a factor of $\Theta(s)$. By the NP-hardness of the problem of approximating Max-Clique by Zuckerman [31], for any constant $\delta > 0$, the length of a LCS of arbitrary number of input sequences of length n each, cannot be approximated within an $n^{1-\delta}$-factor in polynomial time unless P=NP. However, the result of Jiang and Li [18] is only true for alphabets of size $\Omega(n)$. For smaller alphabets (even for size sublinear in n) we do not know any such hardness result. Jiang and Li [18] conjectured that Multi-LCS for even binary alphabet is MAX-SNP-hard (see [26] for the definition of MAX-SNP-hardness). To the best of our knowledge no progress has been done so far on the direction of showing any conditional hardness for smaller alphabets. On the other hand, it is very easy to get a $1/|\Sigma|$-approximation algorithm for the Multi-LCS problem over any alphabet Σ. The algorithm just outputs the best subsequence among the subsequences of the same symbol.

In this paper, we make a significant progress towards showing hardness of approximation of Multi-LCS by refuting the existence of a polynomial time constant factor approximation algorithm under the Exponential Time Hypothesis (ETH).

▶ **Theorem 1.** *There exists a growing function $f(n) = n^{o(1)}$ such that assuming ETH, there is no polynomial time $\frac{1}{f(n)}$-factor approximation algorithm for the Multi-LCS problem over $n^{o(1)}$-sized alphabet.*

This rules out *any* efficient poly-logarithmic factor approximation algorithm for the Multi-LCS problem over any $n^{o(1)}$-sized alphabet. We show the above theorem by providing a polynomial time reduction from the well-studied *densest k-subgraph* problem with *perfect completeness* and its gap version γ-DkS (for the definition see Section 2).

▶ **Theorem 2.** *Let $\frac{k}{n} = \frac{\beta(n)}{\gamma(n)}$ for $\beta < \gamma \leq 1$. If there is no polynomial time algorithm that solves $(\gamma^2/4)$-DkS(k, n), then there is no polynomial time algorithm that solves 2γ-approximate Multi-LCS problem over some alphabet of size $O(\frac{1}{\beta^6})$.*

The above reduction together with the ETH-based hardness result for the densest k-subgraph problem given by Manurangsi [24] implies Theorem 1. We refer to Appendix 1.2 for the previous works related to the LCS problem and the densest k-subgraph problem.

1.1 Techniques

Our reduction starts with the reduction from the Max-Clique problem to Multi-LCS given by [18]. Given a graph G on n vertices the reduction outputs a Multi-LCS instance $\mathcal{I}$ over an alphabet $\{a_1, a_2, \ldots, a_n\}$ of size n with $2n$ strings. The reduction has a guarantee that the maximum LCS size of $\mathcal{I}$ is *equal* to the size of the maximum clique in G.

A natural way to reduce the alphabet size is to replace each symbol a_i in a string with a string $S_i \in \Sigma^m$ over a smaller alphabet Σ. Let us denote this new instance by $\mathcal{I}'$. The hope is that the only way to get a large LCS in $\mathcal{I}'$ is to match the corresponding strings whenever the respective symbols in $\mathcal{I}$ are matched. But this wishful thinking is not true when the alphabet size is much smaller than the original alphabet size as one might get a large common subsequence by matching parts of strings S_i, S_j corresponding to the different symbols a_i, a_j in the original strings.

We get away with this issue by using a special collection of strings $\{S_1, S_2, \ldots, S_n\}$ with the guarantee that for every pair $i \neq j$, $\mathrm{LCS}(S_i, S_j)$ is much smaller than m. We can construct such a set deterministically by using the known deterministic construction of the so called *long-distance synchronization strings* [14, 9]. There is also a much simpler randomized construction (see Theorem 11). It is easy to see that if the original strings have a LCS of size t, then the new Multi-LCS instance $\mathcal{I}'$ over alphabet Σ has an LCS of size at least tm.

The interesting direction is to prove the converse i.e., if the LCS of $\mathcal{I}'$ is large then the LCS of $\mathcal{I}$ is also large. We do not know if this is true in general. So we rely on the starting problem of Max-Clique from which the instance $\mathcal{I}$ (and hence $\mathcal{I}'$) was created. We show that if $\mathcal{I}'$ has large LCS, then we can find a large subgraph of G which has a non trivial density (instead of finding a large clique). Thus, the reduction relies on hardness of approximation of the DkS problem with *perfect completeness*. Then we use the result of Manurangsi [24] which shows that given a graph G with a guarantee that there is a clique of size k, there is no polynomial time algorithm which finds a subgraph of G of size k with density at least $\gamma(n)$ for some $\gamma(n) = o(n)$, assuming the ETH.

1.2 Related works

1.2.1 Results on LCS problem

Finding LCS between two strings is an important problem in computer science. Wagner and Fischer [30] gave a quadratic time algorithm, which is in fact prototypical to dynamic programming. The running time was later improved to (slightly) sub-quadratic, more specifically $O(\frac{n^2 \log\log n}{\log^2 n})$ [25, 12]. Abboud, Backurs and Williams [2] showed that a truly

sub-quadratic algorithm ($O(n^{2-\varepsilon})$ for some $\varepsilon > 0$) would imply a $2^{(1-\delta)n}$ time algorithm for CNF-satisfiability, contradicting the Strong Exponential Time Hypothesis (SETH). They in fact showed that for m input strings an algorithm with running time $O(n^{m-\varepsilon})$ would refute SETH. Abboud et al. [3] later further strengthened the barrier result by showing that even shaving an arbitrarily large polylog factor from n^2 would have the plausible, but hard-to-prove, consequence that NEXP does not have non-uniform NC^1 circuits. In case of approximation algorithm for LCS over arbitrarily large alphabets a simple sampling based technique achieves $O(n^{-x})$-approximation in $O(n^{2-2x})$ time. Very recently, an $O(n^{-0.497956})$ factor approximation (breaking $O(\sqrt{n})$ barrier) linear time algorithm is provided by Hajiaghayi et al. [15]. For binary alphabets another very recent result breaks 1/2-approximation factor barrier in subquadratic time [29]. (Note, $1/|\Sigma|$-approximation over any alphabet Σ is trivial.) The only hardness (or barrier) results for approximating LCS in subquadratic time are presented in [1, 4].

For the general case (which we also refer as Multi-LCS), when the number of input strings is unrestricted, the decision version of the problem is known to be NP-complete [23] even for the binary alphabet. The problem remains NP-complete even with further restriction like bounded *run-length* on input strings [7]. As cited earlier, Jiang and Li [18] (along with the result of Zuckerman [31]) showed that for every constant $\delta > 0$, there is no polynomial time algorithm that achieves $n^{1-\delta}$-approximation factor, unless P=NP. One interesting aspect of the reduction in [18] is that in any input string any particular symbol appears at most twice. It is worth mentioning that if we restrict ourselves to the input strings where a symbol appears exactly once, then we can find a LCS in polynomial time. The algorithm is just an extension of the dynamic programming algorithm that finds a longest increasing subsequence of an input sequence. It is also not difficult to show that the decision version of the Multi-LCS problem with the above restrictions on the input strings can be solved even in non-deterministic logarithmic space. To see this consider a LCS as a certificate. Then the verification algorithm makes single pass on the certificate, and check whether every two consecutive symbols in the certificate appears in the same order in all the input strings. Clearly, the above verification algorithm uses only logarithmic space. Since we know that each symbol appears exactly once in a string, the above verification algorithm correctly decides whether the given certificate is a valid LCS or not.

1.2.2 Hardness results related to densest k-subgraph problem

Our starting point of the reduction is the hardness of approximating the densest k-subgraph problem. In the densest k-subgraph problem (DkS), we are given a graph $G(V, E)$ and an integer $1 \leq k \leq |V|$. The task is to find a subgraph of G of size k with maximum density. Various approximation algorithms are known for DkS [21, 10], and the current best known is by [6] which gives $n^{1/4+\varepsilon}$-approximation algorithm for any constant $\varepsilon > 0$.

A special case of DkS is when it is guaranteed that G has a clique of size k and the task is to find a subgraph of size[1] k with density at least γ for $0 < \gamma \leq 1$. In this *perfect completeness* case, Feige and Seltser [11] gave an algorithm which finds a k sized subgraph with density $(1 - \varepsilon)$ in time $n^{O((1+\log \frac{n}{k})/\varepsilon)}$.

There are several inapproximability results known for DkS based on worst-case assumptions. Khot [19] ruled out a PTAS assuming NP $\not\subseteq$ BPTIME $(2^{n^{\varepsilon}})$ for some constant $\varepsilon > 0$. Raghavendra and Steurer [28] showed that DkS is hard to approximate to within any constant ratio assuming the Unique Games Conjecture where the constraint graph satisfies a small set expansion property.

[1] Note, here *size* of a subgraph refers to the number of vertices present in that subgraph.

Assuming the Exponential Time Hypothesis, Braverman et al. [8], showed that for some constant $\varepsilon > 0$, there is no polynomial time algorithm which when given a graph with a k-clique finds a k sized subgraph with density $(1-\varepsilon)$. This result is significantly improved by Manurangsi [24] in which he showed that assuming ETH, no polynomial time algorithm can distinguish between the cases when G has a clique of size k and when every k sized subgraph has density at most $n^{-1/(\log\log n)^c}$ for some constant $c > 0$.

2 Preliminaries

Notations

We use $[n]$ to denote the set $\{1, 2, \cdots, n\}$. For any string S we use $|S|$ to denote its length. By abuse of notation, for any set V we also use the notation $|V|$ to denote the size of V. For any string S of length n and two indices $i, j \in [n]$, $S[i, j]$ denotes the substring of S that starts at index i and ends at index j. We use $\alpha(n), \beta(n), \gamma(n)$ to denote that α, β, γ are allowed to depend on n.

2.1 Longest Common Subsequence

Given m sequences $S_1, \ldots, S_m$ of length n over an alphabet Σ, the longest common subsequence is the longest sequence S such that $\forall i \in [m], S$ is a subsequence of S_i.

We will refer to the computational problem of finding or deciding the length of LCS as a Multi-LCS problem. In this paper, we consider the decision variant of this problem: Given an integer $\ell \leq n$, we have to decide whether LCS has a length greater than equal to ℓ, or less than ℓ. For the approximation, we consider the following gap-version of this problem.

▶ **Problem 3.** For any $0 < \kappa < 1$, the κ-approximate Multi-LCS problem is defined as: Given sequences $S_1, \ldots, S_m$ of length n over an alphabet Σ and an integer ℓ, the goal is to distinguish between the following two cases

- YES instance: A LCS of $S_1, \ldots, S_m$ has length greater than or equal to ℓ.
- NO instance: A LCS of $S_1, \ldots, S_m$ has length less than $\kappa \cdot \ell$.

We use the following definition of alignment.

▶ **Definition 4** (Alignment). *Given two strings S_1 and S_2 of lengths n and m respectively, alignment σ is a function from $[n]$ to $[m] \cup \{*\}$ which satisfies $\forall i \in [n]$, if $\sigma(i) \neq *$ then $S_1[i] = S_2[\sigma(i)]$ and for any i and j if $\sigma(i) \neq *, \sigma(j) \neq *$ then for $i > j$, $\sigma(i) > \sigma(j)$.*

For an alignment σ between two strings S_1 and S_2 we say σ *aligns* some subsequence $T_1 = S_1[i_1]S_1[i_2]\cdots S_1[i_{\ell_1}]$ of S_1 with some subsequence $T_2 = S_2[j_1]S_2[j_2]\cdots S_2[j_{\ell_2}]$ of S_2 if and only if for all $p \in [\ell_1]$, $\sigma(i_p) \in \{j_1, j_2, \cdots, j_{\ell_2}\}$.

2.2 Exponential Time Hypothesis

The Exponential Time Hypothesis (ETH) was introduced by Impagliazzo and Paturi [17]. It refutes the possibility of getting much faster algorithm to decide satisfiability of a 3-CNF formula (also referred as 3-SAT problem) than that by the trivial brute force method.

▶ **Hypothesis 5** (ETH). *There is no $2^{o(n)}$ time algorithm for the 3-SAT problem over n variables.*

2.3 Densest k-Subgraph problem and related hardness results

For any graph, the density is defined as the ratio of the number of edges present in it and the number of edges in any complete graph of the same size. So given a graph $G = (V, E)$, the density of G is $\frac{2|E|}{|V|^2-|V|}$.

The Densest k-Subgraph (DkS) problem is the following: Given a graph G on n vertices and a positive integer $k \leq n$, the goal is to find a subgraph of G with k vertices which has maximum density.

In this paper we will consider the following gap-version of densest k-subgraph, which in the literature is sometimes referred as densest k-subgraph with *perfect completeness*.

▶ **Problem 6.** For any $\gamma \leq 1$, γ-DkS(k, n) is defined as: Given a graph G on n vertices and a positive integer $k \leq n$, the goal is to distinguish between the following two cases

- YES instance: There exists a clique of size k.
- NO instance: All subgraphs of size k have density at most γ.

We say that an algorithm solves γ-DkS(k, n) if given any input it can distinguish whether the input is a YES instance or a NO instance. If the algorithm is randomized then it should succeed with probability at least 2/3.

In this paper we use the following hardness result by Manurangsi [24].

▶ **Theorem 7** ([24])**.** *There exists a constant $c_0 > 0$ such that assuming the Exponential Time Hypothesis, for all constants $\varepsilon > 0$, there is no polynomial time algorithm for γ-DkS(k, n) where $\gamma = n^{-O\left(\frac{1}{(\log\log n)^{c_0}}\right)}$ and $\frac{k}{n} \in \left[n^{-\varepsilon}, n^{-\Omega\left(\frac{1}{\log\log n}\right)}\right]$.*

3 Reduction

In this section we provide a reduction from the densest k-subgraph problem to the problem of approximating Multi-LCS and prove Theorem 2. Note that, Theorem 2 and Theorem 7 together immediately imply Theorem 1 by plugging $\gamma(n) = n^{-O\left(\frac{1}{(\log\log n)^{c_0}}\right)}, \beta(n) = \gamma(n)^2$.

▶ Remark 8. If we want to get the hardness of Multi-LCS for a constant sized alphabet using Theorem 2 then k must be $\Omega(n)$. However, when $k = \Omega(n)$ Theorem 7 does not imply any hardness result. In fact, when $k = \Omega(n)$, there is a polynomial time algorithm for $(1 - \varepsilon)$-DkS(k, n) for any constant $\varepsilon > 0$ [11]. Therefore our reduction will not give any hardness for constant sized alphabet. However, if one can improve Theorem 7 for $k/n = 1/\text{poly}(\log n)$ and $\gamma(n) = 1/\text{poly}(\log n)$, then our main reduction in Theorem 2 will imply Multi-LCS hardness for $\text{poly}(\log n)$ sized alphabet!

Our reduction involves two steps: First, we use the reduction from the Max-Clique problem to the Multi-LCS problem over large alphabet given in [18]. Next we perform alphabet reduction by replacing each character by a "short" string over a small-sized alphabet.

Revisiting the reduction from Max-Clique to Multi-LCS

We first recall the reduction from [18]. We are given a graph $G = (V, E)$ on n vertices and an integer $k \leq n$. Fix an arbitrary labeling on the vertices of V as $v_1, \ldots, v_n$. For every vertex v_i, partition its neighbors into two subsets: $\mathcal{N}_<(v_i)$ contains all the neighboring vertices v_j with $j < i$; and $\mathcal{N}_>(v_i)$ contains all the neighboring vertices v_j with $j > i$.

Consider an alphabet Σ containing a separate symbol for each vertex. We use v_i to denote both the vertex and its corresponding symbol in Σ. Now for each vertex $v_i \in V$, construct the following two strings X_i and X'_i

$$X_i = v_1 \dots v_{i-1} v_{i+1} \dots v_n v_i v_{i_r} \dots v_{i_s} \text{ and } X'_i = v_{i_p} \dots v_{i_q} v_i v_1 \dots v_{i-1} v_{i+1} \dots v_n$$

where $\mathcal{N}_>(v_i) = \{v_{i_r}, \cdots, v_{i_s}\}$ with $i_r < \cdots < i_s$, and $\mathcal{N}_<(v_i) = \{v_{i_p}, \cdots, v_{i_q}\}$ with $i_p < \cdots < i_q$. The following proposition is immediate from the above construction.

▶ **Proposition 9** ([18]). *If there is a clique of size c in G, then there is a common subsequence of $X_1, \cdots, X_n, X'_1, \cdots, X'_n$ of length c.*

The converse has also been shown in [18].

▶ **Proposition 10** ([18]). *For any common subsequence S of $X_1, \cdots, X_n, X'_1, \cdots, X'_n$, all the v_i's present in S form a clique in G.*

The proofs of these propositions follow from the facts that any common subsequence is of the form $v_{i_1}, v_{i_2}, \dots, v_{i_t}$ where $i_1 < i_2 < \dots < i_t$ and that there must be an edge between v_{i_j} and $v_{i_{j'}}$ for $1 \le j < j' \le t$.

Reducing the size of the alphabet

For some parameter $\alpha(n) < 1$, let $\{S_1, \dots, S_n\}$ be a set of strings of length m over some alphabet Σ' such that: for all $i \neq j$ $|LCS(S_i, S_j)| \le \alpha m$. We will fix the value of m and $|\Sigma'|$ later. The following theorem (Theorem 1 of [20]) shows that if we pick strings from Σ'^m uniformly at random then for $|\Sigma'| = O(1/\alpha^2)$, with high probability the sampled strings will satisfy the above desired property.

▶ **Theorem 11** ([20]). *For every $\varepsilon > 0$ there exists $c > 0$ such that for large enough sized alphabet Σ' for any m if two strings S_1, S_2 are picked uniformly at random from Σ'^m then*

$$Pr\left[\left||LCS(S_1, S_2)| - \frac{2m}{\sqrt{|\Sigma'|}}\right| \ge \varepsilon \frac{2m}{\sqrt{|\Sigma'|}}\right] \le e^{-cm/\sqrt{|\Sigma'|}}.$$

Now by suitably choosing ε, m the following lemma directly follows from a union bound over every pair of n chosen strings.

▶ **Lemma 12.** *For any $\alpha \in (0,1)$, and $n \in \mathbb{N}$ there exists an alphabet Σ' of size $O(\alpha^{-2})$ such that for any $m \ge c\alpha^{-1} \log n$ (for some suitably chosen constant $c > 0$), if we choose a set of strings $S_1, \cdots, S_n$ uniformly at random from Σ'^m then with probability at least $1 - 1/n$ for each $i \neq j$, $|LCS(S_i, S_j)| \le \alpha m$.*

The above lemma gives us a randomized reduction. However we can deterministically find such a collection (with a slight loss in the parameters) using the known construction of *synchronization strings*. The proof of the following Lemma is deferred to Appendix A.

▶ **Lemma 13.** *For any $\alpha \in (0,1)$, and $n \in \mathbb{N}$ there exists an alphabet Σ' of size $O(\alpha^{-3})$ such that for any $m > 2\alpha^{-2} \log n$, there is a deterministic construction of a set of strings $S_1, \cdots, S_n \in \Sigma'^m$ such that for each $i \neq j$, $|LCS(S_i, S_j)| \le \alpha m$. Moreover, all the strings can be generated in time $O(\alpha^{-2} nm)$.*

▶ Remark 14. One advantage of using the randomized construction is the alphabet size (as well as the length of strings); randomized construction has only a quadratic loss whereas the deterministic construction has a cubic loss in the alphabet size. However this will not matter much for the parameters we need to prove our main theorem.

Now let us continue with the description of our reduction. We replace each $v_j \in \Sigma$ by the string S_j. After the replacement we get the following two strings Y_i and Y'_i respectively from X_i and X'_i.

$$Y_i = S_1 \dots S_{i-1} S_{i+1} \dots S_n S_i S_{i_r} \dots S_{i_s} \text{ and } Y'_i = S_{i_p} \dots S_{i_q} S_i S_1 \dots S_{i-1} S_{i+1} \dots S_n$$

Note, Y_i and Y'_i's are over the alphabet Σ'. For notational convenience we use $S_{\mathcal{N}_{>i}}$ to denote the substring $S_{i_r} \dots S_{i_s}$, and $S_{\mathcal{N}_{<i}}$ to denote the substring $S_{i_p} \dots S_{i_q}$. From now on, for simplicity, we will refer to these S_i's as *blocks*. Note, due to deterministic construction of strings S_i's by Lemma 13 our whole reduction is deterministic and polynomial time.

It follows directly from Proposition 9 that:

▶ **Lemma 15** (Completeness). *If graph G is a YES instance of $\frac{\gamma^2}{4}$-DkS (with clique of size k), then a LCS of $Y_1, \dots, Y_n, Y'_1, \dots, Y'_n$ is of length at least km.*

We devote the rest of this section to proving the soundness of our reduction.

▶ **Lemma 16** (Soundness). *Let $\alpha \in (0, 1/8)$ and $\beta = \sqrt{8\alpha}$. If graph G is a NO instance of $\frac{\gamma^2}{4}$-DkS (every subgraph of size k has density less than $\frac{\gamma^2}{4}$), then a LCS of $Y_1, \dots, Y_n, Y'_1, \dots, Y'_n$ has length at most $2\beta mn$.*

3.1 Proof of Soundness

Let L be an (arbitrary) LCS of $Y_1, \cdots, Y_n, Y'_1, \cdots, Y'_n$ of size greater than $2\beta mn$. By the construction $Y_n = S_1 \dots S_n$ (since $\mathcal{N}_>(v_n) = \emptyset$). So we can partition the subsequence L as $Z_1, \cdots, Z_n$ where $\forall i \in [n]$ Z_i is a subsequence of S_i. (Z_i can be an empty string). Now consider all the Z_i of length at least βm, and let $\mathcal{W}$ denote the set of all such Z_i's, i.e., $\mathcal{W} = \{Z_i \mid |Z_i| \geq \beta m\}$. Suppose L_1 is the string formed by removing all $Z_i \notin \mathcal{W}$ from L. Clearly, $|L_1| \geq |L| - \beta mn \geq \beta mn$.

For all $i, j \in [n]$ such that $i < j$, define $C[i, j]$ as: $C[i, j] := \{Z_t \in \mathcal{W} \mid i \leq t \leq j\}$. Note, $\mathcal{W} = C[1, n]$. Next we show that either the size of $C[1, n]$ is small or there exists a subgraph in G which has large density.

Let us consider the set of vertices $V_H := \{v_t | Z_t \in \mathcal{W}\}$. So $|V_H| = |\mathcal{W}| \geq \frac{|L|}{m} - \beta n \geq \beta n$. If we could show that the subgraph H of G induced by the set of vertices V_H has high density (ideally, a clique), then that will imply Lemma 16.

Now consider an (arbitrary) alignment between L_1 and $Y_1, \cdots, Y_n, Y'_1, \cdots, Y'_n$. Let us denote the alignment between L_1 and Y_i (Y'_i) by σ_i (σ'_i). From now on whenever we will talk about alignment we will refer to these particular alignments (σ_i or σ'_i depending on strings under consideration) without specifying them explicitly. Consider a $Z_t \in \mathcal{W}$. We say Z_t is ε-*aligned* (for some $\varepsilon \in [0, 1]$) with some substring S' of some Y_i (or Y'_i) if and only if either the first or the last ε fraction of symbols of Z_t is aligned by the alignment σ'_i (or σ'_i) with some subsequence of S'. Throughout this proof we will set $\varepsilon = 1/2$. Note that, if we partition Y_i into (any) two parts Y_i^l and Y_i^r then Z_i is 1/2-*aligned* to at least one of Y_i^l and Y_i^r, and this justifies our setting of parameter ε.

By following the argument of the proof of Proposition 10 given in [18], it is possible to show that if σ aligns all Z_t with some subsequence of S_t in all strings Y_i (and Y'_i), then the subgraph H induced by vertices in V_H has high density (actually forms a clique). Unfortunately we do not know whether all the Z_t's are aligned with their corresponding S_t's in all the Y_i's (and Y'_i's). Following are the different cases of mapping $Z_i \in \mathcal{W}$ with Y_i:

1. Z_i is 1/2-*aligned* with the substring $S_1 \dots S_{i-1}$ of Y_i.
2. Z_i is 1/2-aligned with $S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ of Y_i and there exists a $j > i$ such that a symbol of Z_j in L_1 is aligned with some symbol of S_j in the substring $S_{i+1} \dots S_n S_i$.
3. Z_i is 1/2-aligned with the substring $S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ in Y_i and there exists no $j > i$ such that a symbol of $Z_j \in \mathcal{W}$ is aligned with some symbol of S_j in the substring $S_{i+1} \dots S_n$.

Similarly, we will also consider the mapping with Y_i''s. We will categorize first and second case as *sparse case* and the third one as the *dense case*. Next we analyze these cases.

3.1.1 Sparse Case: Improper mapping leads to small LCS locally

Let us recall that $Y_i = S_1 \dots S_{i-1} S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ and $Y_i' = S_{\mathcal{N}_{<i}} S_i S_1 \dots S_{i-1} S_{i+1} \dots S_n$. The next two claims demonstrate that if Z_i is not mapped to S_i in Y_i (or Y_i') then there is a portion $C[j,i]$ (or $C[i,j]$) in L_1 such that $\frac{|C[j,i]|}{i-j}$ (or $\frac{|C[i,j]|}{j-i}$) is small, i.e., that portion of L_1 is "sparse" with respect to the number of Z_t blocks present in it.

▷ Claim 17. If $Z_i \in \mathcal{W}$ is 1/2-aligned with the substring $S_1 \dots S_{i-1}$ of Y_i (by the alignment σ_i), then there exists a $j < i$ such that $|C[j,i]| \leq \frac{2\alpha}{\beta}(i-j+1)$. Similarly, if $Z_i \in \mathcal{W}$ is 1/2-aligned with the substring $S_{i+1} \dots S_n$ of Y_i' (by the alignment σ_i'), then there exists a $j > i$ such that $|C[i,j]| \leq \frac{2\alpha}{\beta}(j-i+1)$.

Proof. Suppose Z_i is 1/2-aligned with $S_1 \dots S_{i-1}$ of Y_i. Let j be the largest index less than i such that a symbol in Z_j is aligned (by σ_i) with some symbol in S_j in Y_i (if there does not exist such a j then take $j = 0$). Note, by the definition of 1/2-alignment at least first $\beta m/2$ symbols of Z_i are mapped (by σ_i) in $S_1 \dots S_{i-1}$. Recall, the definition of 1/2-alignment ensures the mapping of the first or the last half fraction of symbols. However in this case if Z_i's last $\beta m/2$ symbols are mapped in $S_1 \dots S_{i-1}$ then the whole Z_i is actually mapped in $S_1 \dots S_{i-1}$, which is even stronger than what we state.

By the properties of strings S_k's specified in Lemma 13, the first $\beta m/2$ symbols of Z_i require at least $\frac{\beta}{2\alpha}$ blocks from $\{S_j, S_{j+1}, \dots, S_{i-1}\}$ to map completely (see Figure 1).

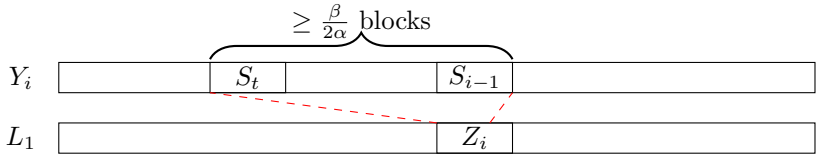


Figure 1 Z_i is 1/2-aligned with $S_1 \dots S_{i-1}$ where $t > j$.

Similarly each element of $C[j+1, i-1]$ also requires at least $\frac{\beta}{\alpha}$ blocks from $\{S_j, S_{j+1}, \dots, S_{i-1}\}$. However any two $Z_p, Z_{p+1} \in C[j+1, i]$ may share a block (more specifically, the last block used for Z_p and the first block used for Z_{p+1}) for mapping. So, we get

$$\frac{\beta}{2\alpha} + (\frac{\beta}{\alpha} - 1)|C[j+1, i-1]| \leq i - j \Rightarrow \frac{\beta}{2\alpha}|C[j+1, i]| \leq i - j.$$

Note, $\frac{\beta}{\alpha} - 1 \geq \frac{\beta}{2\alpha}$ as $\alpha \leq 1/8$ (recall, $\beta = \sqrt{8\alpha}$), and $C[j+1, i-1] \cup \{Z_i\} = C[j+1, i]$.

Similarly, suppose Z_i is 1/2-aligned with $S_{i+1} \dots S_n$ of Y_i'. Let j be the smallest index greater than i such that a symbol of Z_j is aligned (by σ_i') with some symbol of S_j in Y_i' (if there does not exist any j then take $j = n+1$). Using an argument similar to the above, we get

$$\frac{\beta}{2\alpha} + (\frac{\beta}{\alpha} - 1)|C[i+1, j-1]| \leq j - i \Rightarrow \frac{\beta}{2\alpha}|C[i, j-1]| \leq j - i. \quad \triangleleft$$

▷ Claim 18. Suppose (by the alignment σ_i) $Z_i \in \mathcal{W}$ is 1/2-aligned with $S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ of Y_i, and there exists a $j > i$ such that a symbol of Z_j in L_1 is aligned with some symbol of S_j in the substring $S_{i+1} \dots S_n S_i$. Then there exists r such that $i < r \leq j$ and $|C[i, r-1]| \leq \frac{2\alpha}{\beta}(r-i)$.

Similarly, suppose (by the alignment σ'_i) $Z_i \in \mathcal{W}$ is 1/2-aligned with $S_{\mathcal{N}_{<i}} S_i S_1 \dots S_{i-1}$ of Y'_i, and there exists a $j < i$ such that a symbol of Z_j in L_1 is aligned with some symbol of S_j in the substring $S_i S_1 \dots S_{i-1}$. Then there exists r such that $j \leq r < i$ and $|C[r+1, i]| \leq \frac{2\alpha}{\beta}(i-r)$.

Proof. Suppose Z_i is 1/2-aligned with $S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ of Y_i and there exists a $j > i$ such that a symbol of Z_j in L_1 is aligned (by σ_i) with some symbol of S_j in the substring $S_{i+1} \dots S_n S_i$. Let us choose r to be the smallest j with the above condition. By the argument used in the proof of Claim 17, Z_i requires at least $\frac{\beta}{2\alpha}$ blocks from $\{S_{i+1}, S_{i+2}, \cdots, S_r\}$, and every element in $C[i+1, r-1]$ requires at least $\frac{\beta}{\alpha}$ blocks from $\{S_{i+1}, S_{i+2}, \cdots, S_r\}$. Again, any two $Z_p, Z_{p+1} \in C[i, r-1]$ may share a block (more specifically, the last block used for Z_p and the first block used for Z_{p+1}) for mapping. So we get

$$\frac{\beta}{2\alpha} + |C[i+1, r-1]|(\frac{\beta}{\alpha} - 1) \leq r - i \Rightarrow \frac{\beta}{2\alpha}|C[i, r-1]| \leq r - i.$$

Similarly, suppose Z_i is 1/2-aligned with $S_{\mathcal{N}_{<i}} S_i S_1 \dots S_{i-1}$ of Y'_i and there exists a $j < i$ such that a symbol of Z_j in L_1 is aligned (by σ'_i) with some symbol of S_j in the substring $S_i S_1 \dots S_{i-1}$. Let us choose r to be the largest j with the above condition. Then we get

$$\frac{\beta}{2\alpha} + |C[r+1, i-1]|(\frac{\beta}{\alpha} - 1) \leq i - r \Rightarrow \frac{\beta}{2\alpha}|C[r+1, i]| \leq i - r. \quad \triangleleft$$

3.1.2 Dense Case: Proper mapping implies large number of neighbors

Recall that $V_H = \{v_t \mid Z_t \in \mathcal{W}\}$. For each $v_i \in V_H$ further define $V_H^{>i} := \{v_t \in V_H \mid t > i\}$ and $V_H^{<i} := \{v_t \in V_H \mid t < i\}$. The next two claims show that if Z_i is aligned with S_i in Y_i and Y'_i then "most" of the vertices in V_H are connected to (i.e., neighbors of) the vertex v_i. This eventually helps us to show that density of H is high.

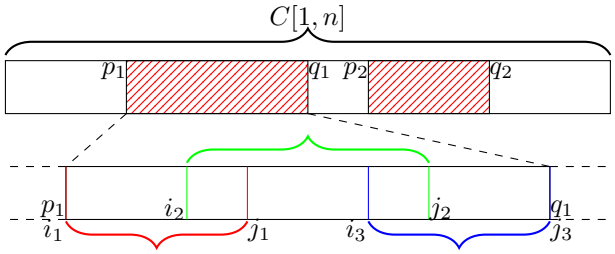
▷ Claim 19. Suppose (by the alignment σ_i) $Z_i \in \mathcal{W}$ is 1/2-aligned with $S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ in Y_i, and there exists no $j > i$ such that a symbol of $Z_j \in \mathcal{W}$ is aligned with some symbol of S_j in the substring $S_{i+1} \dots S_n$. Then

$$|V_H^{>i} \bigcap \mathcal{N}_>(v_i)| + \frac{\beta}{2\alpha}|V_H^{>i} \setminus \mathcal{N}_>(v_i)| \leq 2(n-i) + 1.$$

Proof. Z_i is 1/2-aligned with $S_{i+1} \dots S_n S_i S_{\mathcal{N}_{>i}}$ of Y_i. So to align all $Z_r \in C[i+1, n]$ (note, $|C[i+1, n]| = |V_H^{>i}|$) at most $2(n-i)+1$ blocks of S_p's are available. Since for no $j > i$ a symbol of $Z_j \in \mathcal{W}$ is aligned with some symbol of S_j in $S_{i+1} \dots S_n$, each Z_r such that $v_r \in V_H^{>i} \setminus \mathcal{N}_>(v_i)$ requires at least $\frac{\beta}{\alpha}$ blocks of S_p's to map. Any two Z_r, Z_{r+1} such that $v_r, v_{r+1} \in V_H^{>i} \setminus \mathcal{N}_>(v_i)$ may share a block (more specifically, the last block used for Z_p and the first block used for Z_{p+1}) for mapping. Recall for our choice of parameters α, β, $\frac{\beta}{\alpha} - 1 \geq \frac{\beta}{2\alpha}$. So we get

$$|V_H^{>i} \bigcap \mathcal{N}_>(v_i)| + \frac{\beta}{2\alpha}|V_H^{>i} \setminus \mathcal{N}_>(v_i)| \leq 2(n-i) + 1. \quad \triangleleft$$

Similarly, we consider the mapping of Z_i in the string Y'_i.



Shaded region is included in $\mathcal{T}$

Considering $s = 3$, (i_1, j_1),(i_2, j_2),(i_3, j_3) is a series of pairs to cover $C[p_1, q_1]$ where $i_1 = p_1$ and $j_3 = q_1$.

Figure 2 $\mathcal{T}$ as a union of disjoint subsets.

▷ Claim 20. Suppose (by the alignment σ'_i) $Z_i \in \mathcal{W}$ is 1/2-aligned with $S_{\mathcal{N}_<(i)} S_i S_1 \dots S_{i-1}$ in Y'_i, and there exists no $j < i$ such that a symbol of $Z_j \in \mathcal{W}$ is aligned with some symbol of S_j in the substring $S_1 \dots S_{i-1}$. Then

$$|V_H^{<i} \bigcap \mathcal{N}_<(v_i)| + \frac{\beta}{2\alpha} |V_H^{<i} \setminus \mathcal{N}_<(v_i)| \leq 2i - 1.$$

Proof. Z_i is 1/2-aligned with $S_{\mathcal{N}_<(i)} S_i S_1 \dots S_{i-1}$ of Y'_i. So to align all $Z_r \in C[1, i-1]$ (note, $|C[1, i-1]| = |V_H^{<i}|$), at most $2i-1$ blocks of S_p's are available. Since for no $j < i$ a symbol of $Z_j \in \mathcal{W}$ is aligned with some symbol of S_j in $S_1 \dots S_{i-1}$, each Z_r such that $v_r \in V_H^{<i} \setminus \mathcal{N}_<(v_i)$ requires at least $\frac{\beta}{\alpha}$ blocks of S_p's to map. Any two Z_r, Z_{r+1} such that $v_r, v_{r+1} \in V_H^{<i} \setminus \mathcal{N}_<(v_i)$ may share a block (more specifically, the last block used for Z_p and the first block used for Z_{p+1}) for mapping. Recall for our choice of parameters α, β, $\frac{\beta}{\alpha} - 1 \geq \frac{\beta}{2\alpha}$. So we get

$$|V_H^{<i} \bigcap \mathcal{N}_<(v_i)| + \frac{\beta}{2\alpha} |V_H^{<i} \setminus \mathcal{N}_<(v_i)| \leq 2i - 1.$$ ◁

3.1.3 Removing sparse blocks from LCS

Next we choose a subset of vertices from the set V_H so that the graph induced by that subset has high density. For that purpose we remove the "sparse" portions from the subsequence L_1 in the following way:

1. Initialize an empty set $\mathcal{T}$.
2. For each $Z_i \in \mathcal{W}$ identify the largest $j > i$ such that $\frac{|C[i,j]|}{j-i+1} \leq \frac{2\alpha}{\beta}$, and then add all $Z_k \in C[i, j]$ in the set $\mathcal{T}$. (If no such j exists then do not add anything to $\mathcal{T}$.)
3. Define a new set $\mathcal{W}' = \mathcal{W} \setminus \mathcal{T}$.

Let L_2 be the string formed by removing all $Z_i \notin \mathcal{W}'$ from L_1. Let us also define a set of vertices $V'_H = \{v_t | Z_t \in \mathcal{W}'\}$. (Note, $V'_H \subseteq V_H$.) Now we will argue that the set V_H has not shrunk by much after removing the sparse blocks and each vertex in V'_H has high degree in the subgraph H, which eventually implies that the subgraph H has high density.

▷ Claim 21. $|V'_H| \geq |V_H| - \frac{4\alpha}{\beta} n$.

Proof. Let us consider the set $\mathcal{T}$. We can write $\mathcal{T}$ as a union of disjoint subsets as $\mathcal{T} = C[p_1, q_1] \cup C[p_2, q_2] \cup \dots \cup C[p_r, q_r]$ for some integer $r \in [n]$, such that $\forall_{1 \leq \ell \leq r-1}\ C[q_\ell, p_{\ell+1}] \neq \emptyset$ (see Figure 2).

Now if we could show that for each $\ell \in [r]$, $|C[p_\ell, q_\ell]| \leq \frac{4\alpha}{\beta}(q_\ell - p_\ell)$, then

$$|\mathcal{T}| = \sum_{\ell=1}^{r} |C[p_\ell, q_\ell]| \leq \frac{4\alpha}{\beta} \sum_{\ell=1}^{r} (q_\ell - p_\ell) \leq \frac{4\alpha}{\beta} n$$

where the last inequality is true since $p_1 < q_1 < p_2 < q_2 < \dots < p_r < q_r$. So to conclude the proof of the claim next we show that for all $\ell \in [r]$ $|C[p_\ell, q_\ell]| \leq \frac{4\alpha}{\beta}(q_\ell - p_\ell)$.

It is immediate from the construction of the set $\mathcal{T}$ that there exists a sequence of pair of indices $(i_1, j_1), \cdots, (i_s, j_s)$ (for some positive integer s) where $i_1 = p_\ell$ and $j_s = q_\ell$, such that for all $t \in [s]$ while processing Z_{i_t} we add blocks of $C[i_t, j_t]$ in $\mathcal{T}$, and $C[p_\ell, q_\ell] = \bigcup_{t\in[s]} C[i_t, j_t]$. We can further assume that there exists no $t' \in [s]$ such that $C[i_{t'}, j_{t'}] \subseteq \bigcup_{t\in[s]\setminus\{t'\}} C[i_t, j_t]$. (In words it means that $C[i_1, j_1], \cdots, C[i_s, j_s]$ is a minimal sequence of subsets whose union is $C[i_1, j_s]$.) Due to this assumption we can write that $i_2 \le j_1 \le i_3 \le j_2 \le \cdots \le i_s \le j_{s-1}$ and $\forall t \in [s-2], i_{t+2} \ge j_t + 1$ (see Figure 2). So,

$$\begin{aligned}
|C[p_\ell, q_\ell]| \le \sum_{t=1}^{s} |C[i_t, j_t]| &\le \frac{2\alpha}{\beta} \sum_{t=1}^{s} (j_t - i_t + 1) \\
&= \frac{2\alpha}{\beta}\Big[s + (j_s - i_1) + \sum_{t=1}^{s-1}(j_t - i_{t+1})\Big] \\
&\le \frac{2\alpha}{\beta}\Big[s + (j_s - i_1) + (j_{s-1} - i_2 - (s-2))\Big] \\
&\le \frac{2\alpha}{\beta}\Big[2(j_s - i_1)\Big]
\end{aligned}$$

where second last inequality uses the fact that $\forall t \in [s-2], i_{t+2} \ge j_t + 1$ and last inequality uses the fact that $j_s \ge j_{s-1} + 1$ and $i_2 \ge i_1 + 1$. Hence we conclude that $|C[p_\ell, q_\ell]| \le \frac{4\alpha}{\beta}(q_\ell - p_\ell)$, and this completes the proof. ◁

▷ Claim 22. For each vertex $v_i \in V'_H$, $|V_H \bigcap \mathcal{N}(v_i)| \ge |V_H| - \frac{4\alpha}{\beta} n$.

Proof. By the construction of $\mathcal{W}'$, for each $Z_i \in \mathcal{W}'$ we know that there exists no $j > i$ (or $< i$) such that $\frac{|C[i,j]|}{j-i+1} \le \frac{2\alpha}{\beta}$ (or $\frac{|C[j,i]|}{i-j+1} \le \frac{2\alpha}{\beta}$). Then by Claim 17 and Claim 18 it follows that all $Z_i \in \mathcal{W}'$ satisfy preconditions of both Claim 19 and Claim 20. Otherwise by Claim 17 and Claim 18 we know that there exists a $j > i$ (or $< i$) such that $\frac{|C[i,j]|}{j-i+1} \le \frac{2\alpha}{\beta}$ (or $\frac{|C[j,i]|}{i-j+1} \le \frac{2\alpha}{\beta}$). For $j > i$ when we process Z_i to construct the set $\mathcal{T}$ we add all the blocks of $C[i, j]$, and for $j < i$ when we process Z_j we add all the blocks of $C[j, i]$. So it must be the case that the alignment σ_i between L_1 and Y_i, 1/2-aligns Z_i to the substring $S_{i+1} \dots S_n S_i S_{\mathcal{N}_>{}_i}$ and there exists no $j > i$ such that $Z_j \in \mathcal{W}$ aligns with S_j in the substring $S_{i+1} \dots S_n$. Also, σ'_i 1/2-aligns Z_i to the substring $S_{\mathcal{N}_<{}_i} S_i S_1 \dots S_{i-1}$ and there exists no $j < i$ such that $Z_j \in \mathcal{W}$ aligns with S_j in the substring $S_1 \dots S_{i-1}$. So by Claim 19

$$|V_H^{>i} \bigcap \mathcal{N}_>(v_i)| + \frac{\beta}{2\alpha}|V_H^{>i} \setminus \mathcal{N}_>(v_i)| \le 2(n-i) + 1,$$

and by Claim 20

$$|V_H^{<i} \bigcap \mathcal{N}_<(v_i)| + \frac{\beta}{2\alpha}|V_H^{<i} \setminus \mathcal{N}_<(v_i)| \le 2i - 1.$$

These two claims together imply

$$\begin{aligned}
&|V_H \bigcap \mathcal{N}(v_i)| + \frac{\beta}{2\alpha}|V_H \setminus \mathcal{N}(v_i)| \le 2n \\
\Rightarrow &|V_H \bigcap \mathcal{N}(v_i)| + \frac{\beta}{2\alpha}(|V_H| - |V_H \bigcap \mathcal{N}(v_i)|) \le 2n \\
\Rightarrow &(\frac{\beta}{2\alpha} - 1)|V_H \bigcap \mathcal{N}(v_i)| \ge \frac{\beta}{2\alpha}|V_H| - 2n \\
\Rightarrow &|V_H \bigcap \mathcal{N}(v_i)| \ge |V_H| - \frac{4\alpha}{\beta} n.
\end{aligned}$$

◁

Now we are ready to complete the proof of soundness (Lemma 16).

Proof of Lemma 16. For the sake of contradiction let us assume that the LCS is of size at least $2\beta mn$. Recall, we have already seen that $|V_H| \geq \beta n$. Now we consider the following two cases depending on the size of V_H.

Case 1: (When $|V_H| \leq \frac{\beta}{\gamma}n$). Suppose $|V_H| \leq \frac{\beta}{\gamma}n$ $(= k)$. Let $V' \supseteq V_H$ be an arbitrary set of size exactly $\frac{\beta}{\gamma}n$. Let H' be the subgraph induced by the vertices V'. Using Claim 21 and Claim 22, we can lower bound the density of the subgraph H' by:

$$\frac{\frac{1}{2}\sum_{v\in V'_H}\left(|V_H| - \frac{4\alpha}{\beta}n\right)}{\binom{|V'|}{2}} \geq \frac{\left(\beta - \frac{4\alpha}{\beta}\right)n \cdot \left(\beta - \frac{4\alpha}{\beta}\right)n}{\frac{\beta}{\gamma}n \cdot \frac{\beta}{\gamma}n} \geq \left(\gamma - \frac{4\alpha\gamma}{\beta^2}\right)^2.$$

As we set $\alpha = \beta^2/8$, we get that the density of the subgraph induced by V' is at least $(\gamma/2)^2$.

Case 2: (When $|V_H| > \frac{\beta}{\gamma}n$). If $|V_H| > \frac{\beta}{\gamma}n$, the density of the subgaph H induced by V_H is lower bounded by:

$$\begin{aligned}
\frac{\frac{1}{2}\sum_{v\in V'_H}\left(|V_H| - \frac{4\alpha}{\beta}n\right)}{\binom{|V_H|}{2}} &\geq \frac{|V'_H|\left(|V_H| - \frac{4\alpha}{\beta}n\right)}{|V_H|(|V_H|-1)} \\
&\geq \frac{\left(|V_H| - \frac{4\alpha}{\beta}n\right)^2}{|V_H|^2} \\
&= \left(1 - \frac{4\alpha n}{\beta|V_H|}\right)^2 \\
&\geq (1 - \frac{\gamma}{2})^2 && \text{(since } |V_H| > \frac{\beta}{\gamma}n \text{ and we set } \alpha = \beta^2/8) \\
&\geq (\gamma/2)^2 && \text{(since } \gamma \leq 1).
\end{aligned}$$

Now since density of the subgraph is at least $(\gamma/2)^2$, it follows from the following simple claim that there exists a subgraph of H of size $\frac{\beta}{\gamma}n$ which has density at least $(\gamma/2)^2$.

▷ Claim 23. Suppose a graph $G = (V, E)$ has edge density c, then for any $2 \leq k \leq |V|$, there exists a subgraph of size k with density at least c.

Proof. Let $n = |V|$. Pick a subset $H \subseteq V$ of size exactly k uniformly at random. For a fixed edge e in G, the probability that the edge e is present in the subgraph induced by H is exactly $\frac{\binom{n-2}{k-2}}{\binom{n}{k}}$. Since G has $c \cdot \binom{n}{2}$ edges, by linearity of expectation, the expected number of edges in the subgraph induced by H is equal to $c \cdot \binom{n}{2} \cdot \frac{\binom{n-2}{k-2}}{\binom{n}{k}} = c \cdot \binom{k}{2}$. Therefore, the expected density of the subgraph is exactly equal to c. Hence, by an averaging argument, there exists a subgraph of G of size k with density at least c. ◁

In both the cases, we have shown that there exists a subgraph of size $\frac{\beta}{\gamma}n(= k)$ with density at least $(\gamma/2)^2$, which is a contradiction to the fact that we started with a NO instance of $\frac{\gamma^2}{4}$-DkS$\left(\frac{\beta}{\gamma}n, n\right)$. Therefore in this case, the size of LCS must be at most $2\beta mn$. ◀

Proof of Theorem 2

If there is no polynomial time algorithm to distinguish between the YES and NO instances of $\frac{\gamma^2}{4}$-DkS$\left(\frac{\beta}{\gamma}n, n\right)$, then using Lemma 15 and Lemma 16, it follows that there is no polynomial time algorithm to distinguish between the cases when the LCS of $Y_1, \cdots, Y_n, Y'_1, \cdots, Y'_n$ is of size $\frac{\beta}{\gamma}mn$ vs. $2\beta mn$. Also note that if we use Lemma 13 to construct the strings S_i's then the alphabet size is $O(\alpha^{-3}) = O(\beta^{-6})$. This proves the main theorem.

4 Conclusion

In this paper we show hardness of constant factor approximation of Multi-LCS problem with input of length n over $n^{o(1)}$ sized alphabet assuming the Exponential Time Hypothesis (ETH). This is the first hardness result for approximating Multi-LCS problem for sublinear sized alphabet. To prove our result we provide a reduction from the densest k-subgraph problem with perfect completeness, and then use the known hardness results for the latter problem from [24]. One interesting fact is that if one could show hardness of the γ-DkS(k, n) problem for $k = \Theta(\frac{n}{poly \log n})$ and $\gamma = (\log n)^{-c}$ for some $c > 0$, then due to our reduction that will directly imply constant factor hardness for Multi-LCS over poly-logarithmic sized alphabet under ETH.

References

1. Amir Abboud and Arturs Backurs. Towards hardness of approximation for polynomial time problems. In *8th Innovations in Theoretical Computer Science Conference, ITCS 2017, January 9-11, 2017, Berkeley, CA, USA*, pages 11:1–11:26, 2017.
2. Amir Abboud, Arturs Backurs, and Virginia Vassilevska Williams. Tight hardness results for LCS and other sequence similarity measures. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 59–78, 2015.
3. Amir Abboud, Thomas Dueholm Hansen, Virginia Vassilevska Williams, and Ryan Williams. Simulating branching programs with edit distance and friends: or: a polylog shaved is a lower bound made. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 375–388, 2016.
4. Amir Abboud and Aviad Rubinstein. Fast and deterministic constant factor approximation algorithms for LCS imply new circuit lower bounds. In *9th Innovations in Theoretical Computer Science Conference, ITCS 2018, January 11-14, 2018, Cambridge, MA, USA*, pages 35:1–35:14, 2018.
5. Lasse Bergroth, Harri Hakonen, and Timo Raita. A survey of longest common subsequence algorithms. In Pablo de la Fuente, editor, *Seventh International Symposium on String Processing and Information Retrieval, SPIRE 2000, A Coruña, Spain, September 27-29, 2000*, pages 39–48. IEEE Computer Society, 2000.
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 Leonard J. Schulman, editor, *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 201–210. ACM, 2010.
7. Guillaume Blin, Laurent Bulteau, Minghui Jiang, Pedro J. Tejada, and Stéphane Vialette. Hardness of longest common subsequence for sequences with bounded run-lengths. In *Combinatorial Pattern Matching - 23rd Annual Symposium, CPM 2012, Helsinki, Finland, July 3-5, 2012. Proceedings*, pages 138–148, 2012.
8. Mark Braverman, Young Kun Ko, Aviad Rubinstein, and Omri Weinstein. ETH hardness for densest-k-subgraph with perfect completeness. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1326–1341. SIAM, 2017.

9 Kuan Cheng, Bernhard Haeupler, Xin Li, Amirbehshad Shahrasbi, and Ke Wu. Synchronization strings: Highly efficient deterministic constructions over small alphabets. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2185–2204, 2019.

10 Uriel Feige, David Peleg, and Guy Kortsarz. The dense k-subgraph problem. *Algorithmica*, 29(3):410–421, 2001.

11 Uriel Feige and Michael Seltser. On the densest k-subgraph problem. *Algorithmica*, 29, 1997.

12 Szymon Grabowski. New tabulation and sparse dynamic programming based techniques for sequence similarity problems. *Discrete Applied Mathematics*, 212:96–103, 2016.

13 Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology*. Cambridge University Press, 1997.

14 Bernhard Haeupler and Amirbehshad Shahrasbi. Synchronization strings: explicit constructions, local decoding, and applications. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 841–854, 2018.

15 MohammadTaghi Hajiaghayi, Masoud Seddighin, Saeed Seddighin, and Xiaorui Sun. Approximating LCS in linear time: Beating the $\sqrt{n}$ barrier. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 1181–1200, 2019.

16 D.S. Hirschberg. Recent results on the complexity of common subsequence problems. In *Time Warps, String Edits, and Macromolecules, D. Sankoff and J.B. Kruskal, ed., Addison-Wesley*, pages 323–328, 1983.

17 Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-SAT. *Journal of Computer and System Sciences*, 62(2):367–375, 2001.

18 Tao Jiang and Ming Li. On the approximation of shortest common supersequences and longest common subsequences. *SIAM J. on Computing*, 24(5):1122–1139, 1995.

19 Subhash Khot. Ruling out ptas for graph min-bisection, dense k-subgraph, and bipartite clique. *SIAM Journal on Computing*, 36(4):1025–1071, 2006.

20 Marcos Kiwi, Martin Loebl, and Jiří Matoušek. Expected length of the longest common subsequence for large alphabets. *Advances in Mathematics*, 197(2):480–498, 2005.

21 G Kortsarz and D Peleg. On choosing a dense subgraph. In *Proceedings of the 1993 IEEE 34th Annual Foundations of Computer Science*, pages 692–701. IEEE Computer Society, 1993.

22 S. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man, and Cybernetics*, 8(5):381–389, May 1978.

23 David Maier. The complexity of some problems on subsequences and supersequences. *J. ACM*, 25(2):322–336, April 1978.

24 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. ACM, 2017.

25 William J. Masek and Michael S. Paterson. A faster algorithm computing string edit distances. *Journal of Computer and System Sciences*, 20(1):18–31, 1980.

26 Christos H. Papadimitriou and Mihalis Yannakakis. Optimization, approximation, and complexity classes. *J. Comput. Syst. Sci.*, 43(3):425–440, 1991.

27 Pavel A. Pevzner. Multiple alignment with guaranteed error bounds and communication cost. In *Combinatorial Pattern Matching, Third Annual Symposium, CPM 92, Tucson, Arizona, USA, April 29 - May 1, 1992, Proceedings*, pages 205–213, 1992.

28 Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the forty-second ACM symposium on Theory of computing*, pages 755–764. ACM, 2010.

29 Aviad Rubinstein and Zhao Song. Reducing approximate longest common subsequence to approximate edit distance. In Shuchi Chawla, editor, *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1591–1600. SIAM, 2020.

30 Robert A. Wagner and Michael J. Fischer. The string-to-string correction problem. *J. ACM*, 21(1):168–173, January 1974.
31 David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(6):103–128, 2007.

A Derandomized version of Lemma 12

To achieve deterministic reduction we need to construct the set of strings $S_1, \cdots, S_n$ deterministically in time $poly(n)$. For that purpose we use the notion of *synchronization strings* used in the literature of *insertion-deletion codes* [14, 9].

▶ **Definition 24** (c-long-distance ε-synchronization string). *A string $S \in \Sigma^n$ is called a c-long-distance ε-synchronization string for some parameter $\varepsilon \in (0,1)$, if for every $1 \leq i < j \leq i' < j' \leq n$ with $i' - j \leq n \cdot \mathbb{1}_{(j+j'-i-i')>c\log n}$, $|LCS(S[i,j], S[i',j'])| \leq \varepsilon(j + j' - i - i')$, where $\mathbb{1}_{(j+j'-i-i')>c\log n}$ is the indicator function for $(j + j' - i - i') > c \log n$.*

Note, in the definition of c-long-distance ε-synchronization string in [9] authors used the notion of *edit distance* instead of LCS. More specifically, they specified the edit distance between $S[i,j]$ and $S[i',j']$) is at least $(1-\varepsilon)(|S[i,j]| + |S[i',j']|)$. However both the notions can be used interchangeably since for any two strings S, S', $|LCS(S,S')| = |S| + |S'| - ED(S,S')$, where the edit distance $ED(S,S')$ is defined as the minimum number of insertion and deletion operations required to transform S to S'. One may note that, generally while defining the edit distance we also allow substitution operation. However here we are not allowing substitution operation, and that is why we are able to write the following equivalence between LCS and the edit distance of two strings S, S': $|LCS(S,S')| = |S| + |S'| - ED(S,S')$. We would like to mention that in [9] authors also used this particular version of the edit distance notion (i.e., without substitution operation).

Several constructions of such long-distance synchronization strings are given in [14, 9] with different parameters. However we restate one of the theorems from [9] that we find useful for our purpose.

▶ **Theorem 25** (Rephrasing of Theorem 5.4 of [9]). *For any $n \in \mathbb{N}$ and parameter $\varepsilon \in (0,1)$, there is a deterministic construction of an ε^{-2}-long-distance ε-synchronization string $S \in \Sigma^n$ for some alphabet Σ of size $O(\varepsilon^{-3})$. Moreover, for any $i \in [n]$ the substring $S[i, i + \log n]$ can be computed in time $O(\varepsilon^{-2} \log n)$.*

Now using the above we will provide deterministic construction of set of strings $S_1, \cdots, S_n$ with our desired property.

▶ **Lemma 13.** *For any $\alpha \in (0,1)$, and $n \in \mathbb{N}$ there exists an alphabet Σ' of size $O(\alpha^{-3})$ such that for any $m > 2\alpha^{-2} \log n$, there is a deterministic construction of a set of strings $S_1, \cdots, S_n \in \Sigma'^m$ such that for each $i \neq j$, $|LCS(S_i, S_j)| \leq \alpha m$. Moreover, all the strings can be generated in time $O(\alpha^{-2} nm)$.*

Proof. For a specified α and n, set $\varepsilon = \alpha/2$. Then use the construction from Theorem 25 to get an ε^{-2}-long-distance ε-synchronization string S of length $2nm$, for any $m > \frac{1}{2}\varepsilon^{-2} \log n$. The bound on m is required to satisfy the condition that $(j + j' - i - i') > c \log n$ of Definition 24. (Note, in our case $(j + j' - i - i') = 2m$ and $c = \varepsilon^{-2}$.) Then divide the string S into m length blocks. Finally choose alternate blocks as $S_1, \cdots, S_n$. More specifically, $S_1 = S[1,m], S_2 = S[2m+1, 3m], \cdots, S_n = S[(2n-2)m+1, (2n-1)m]$. Now the bound on $|LCS(S_i, S_j)|$ for any $i \neq j$, directly follows from Definition 24. ◀

On Approximating Degree-Bounded Network Design Problems

Xiangyu Guo
Department of Computer Science and Engineering, University at Buffalo, NY, USA
xiangyug@buffalo.edu

Guy Kortsarz
Department of Computer Science, Rutgers University Camden, NJ, USA
guyk@camden.rutgers.edu

Bundit Laekhanukit
ITCS, Shanghai University of Finance and Economics, China
bundit@sufe.edu.cn

Shi Li
Department of Computer Science and Engineering, University at Buffalo, NY, USA
shil@buffalo.edu

Daniel Vaz
Operations Research Group, TU Munich, Germany
daniel.vaz@tum.de

Jiayi Xian
Department of Computer Science and Engineering, University at Buffalo, NY, USA
jxian@buffalo.edu

Abstract

Directed Steiner Tree (DST) is a central problem in combinatorial optimization and theoretical computer science: Given a directed graph $G = (V, E)$ with edge costs $c \in \mathbb{R}^E_{\geq 0}$, a root $r \in V$ and k terminals $K \subseteq V$, we need to output a minimum-cost arborescence in G that contains an $r \to t$ path for every $t \in K$. Recently, Grandoni, Laekhanukit and Li, and independently Ghuge and Nagarajan, gave quasi-polynomial time $O(\log^2 k / \log\log k)$-approximation algorithms for the problem, which are tight under popular complexity assumptions.

In this paper, we consider the more general Degree-Bounded Directed Steiner Tree (DB-DST) problem, where we are additionally given a degree bound d_v on each vertex $v \in V$, and we require that every vertex v in the output tree has at most d_v children. We give a quasi-polynomial time $(O(\log n \log k), O(\log^2 n))$-bicriteria approximation: The algorithm produces a solution with cost at most $O(\log n \log k)$ times the cost of the optimum solution that violates the degree constraints by at most a factor of $O(\log^2 n)$. This is the first non-trivial result for the problem.

While our cost-guarantee is nearly optimal, the degree violation factor of $O(\log^2 n)$ is an $O(\log n)$-factor away from the approximation lower bound of $\Omega(\log n)$ from the Set Cover hardness. The hardness result holds even on the special case of the *Degree-Bounded Group Steiner Tree* problem on trees (DB-GST-T). With the hope of closing the gap, we study the question of whether the degree violation factor can be made tight for this special case. We answer the question in the affirmative by giving an $(O(\log n \log k), O(\log n))$-bicriteria approximation algorithm for DB-GST-T.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Directed Steiner Tree, Group Steiner Tree, degree-bounded

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.39

Category APPROX

Related Version The arXiv version is available at `https://arxiv.org/abs/1907.11404`.



© Xiangyu Guo, Guy Kortsarz, Bundit Laekhanukit, Shi Li, Daniel Vaz, and Jiayi Xian;
licensed under Creative Commons License CC-BY



Funding X. Guo, S. Li and J. Xian are partially supported by NSF grants CCF-1566356, CCF-1717134, CCF-1844890.
Guy Kortsarz: G. Kortsarz is supported by NSF grants 1540547 and 1910565.
Bundit Laekhanukit: B. Laekhanukit is partially supported by Science and Technology Innovation 2030 – "New Generation of Artificial Intelligence" Major Project No.(2018AAA0100903), NSFC grant 61932002, Program for Innovative Research Team of Shanghai University of Finance and Economics (IRTSHUFE) and the Fundamental Research Funds for the Central Universities and by the 1000-talent award by the Chinese Government.
Daniel Vaz: Daniel Vaz has been supported by the Alexander von Humboldt Foundation with funds from the German Federal Ministry of Education and Research (BMBF).

1 Introduction

Network design is a central problem in combinatorial optimization and computer science. To capture more practical situations, the more general model of network design with *degree-constraints* was suggested in the early 90's [22, 8] and has attracted researchers in both theory and practice for decades. One of the most famous examples is the *Degree-Bounded Minimum Spanning Tree* (DB-MST) problem, which models the problem of designing a multi-casting network in which each node only has enough power to broadcast to a bounded number of its neighbors. This problem has been studied in a sequence of works (see, e.g.,[16, 18, 11, 24]), leading to the breakthrough result of Goemans [11] followed by the work of Singh and Lau [24], which settled the problem by giving an algorithm that outputs a solution with optimum cost, while violating the degree bound by an additive factor of $+1$. Since the works on DB-MST, many efforts have been dedicated to the study the generalizations of the problem: the *Degree-Bounded Steiner Tree* problem, in which the goal is to find a minimum-cost subgraph that connects all the terminals, while meeting the given degree bounds, was studied in [17, 21]. The *Survivable Network Design* problem, where each pair of nodes v, w are required to have at least λ_{vw} edge-disjoint v-w paths, has also been studied in literature; see, e.g., [20, 21].

Recently, degree-bounded network design problems have been studied in the online setting [4, 3, 5]. Besides the standard (also called point-to-point) network design problems, Dehghani et al. [4] also studied the *Degree-Bounded Group Steiner Tree* problem (DB-GST). They gave a negative result, which shows that it is not possible to approximate both cost and weight of the Online DB-GST problem simultaneously, even when the input graph is a star. More specifically, there exists an input demand sequence that forces any algorithm to pay a factor of $\Omega(n)$ either in the cost or in the degree violation. To date, there was no non-trivial approximation algorithm for DB-GST, either in the online or offline setting, and even when all the edges have zero-cost. This was listed as an open problem by Hajiaghayi [14] at the 8th Flexible Network Design Workshop (FND 2016).

In this paper, we study a degree-bounded variant of the classic network design problem, the *Degree-Bounded Directed Steiner Tree* problem (DB-DST). Formally, in DB-DST, we are given an n-vertex directed graph $G = (V, E)$ with costs on edges, a root vertex r, a set of k terminals K, and degree bounds d_v for each vertex v. The goal is to find a minimum-cost rooted tree $T \subseteq G$ that contains a path from the root r to every terminal $t \in K$, while respecting the degree bound, i.e., the out-degree of each vertex v in T is at most d_v. Despite being a classic problem, there was no previous positive result on DB-DST as it is a generalization of DB-GST.

The barriers in obtaining any non-trivial approximation algorithm for DB-GST and DB-DST are similar. Most of the previous algorithms for these two problems either run on the metric closure of the input graph [9, 7, 23], require metric-tree embedding [9, 1, 6] or use height-reduction techniques [25, 2, 13, 10], all of which lose track of the degree of the solution subgraph.

We present a bi-criteria $(O(\log k \log n), O(\log^2 n))$-approximation algorithm for DB-DST that runs in quasi-polynomial-time (see Section 1.1 for the definition), thus solving the open problem of Hajiaghayi [14]. Our technique expands upon the recent result of Grandoni, Laekhanukit and Li [13] for the Directed Steiner Tree problem. We observe that the algorithm in [13] can be easily extended to the problem with degree bounds. Nevertheless, to amend the degree-constrained problem into their framework, we are required to prove a concentration bound for degrees, which is rather non-trivial. Notice that the $O(\log n \log k)$-approximation factor on the cost of the tree is almost tight due to the hardness of $\Omega(\log^{2-\epsilon} n)$ in [15] for Directed Steiner Tree and the slightly improved hardness of $\Omega(\log^2 n/\log\log n)$ in [13].

While our result for DB-DST is (almost) tight on the cost guarantee, the degree violation factor $O(\log^2 n)$ is an $O(\log n)$ factor away from the approximation lower bound of $\Omega(\log n)$ from the Set Cover hardness. To understand if the gap can be reduced, we study the special case of DB-DST obtained from the hardness construction in [15], namely the *Degree-Bounded Group Steiner Tree problem on trees* (DB-GST-T). In this problem, we are given an (undirected) tree $T^\circ = (V^\circ, E^\circ)$ with edge-costs, a root r, k subsets of vertices (called groups) $O_1, \ldots, O_k \subseteq V$ and a degree bound d_v for each vertex $v \in V^\circ$. The goal is to find a minimum-cost subtree $T \subseteq T^\circ$ that joins r to at least one vertex from each group O_t, for every $t \in [k]$, while respecting the degree bound, i.e., the number of children of each vertex v in T is at most d_v. We present an $(O(\log k \log n), O(\log n))$-bicriteria approximation algorithm for DB-GST-T. So, the degree violation of our algorithm is tight and the cost-guarantee is almost tight. This improves upon the $O(\log n \log k, \log n \log k)$-bicriteria approximation algorithm due to Kortsarz and Nutov [19] who observe that the randomized rounding algorithm in [9] also gives a guarantee on degree-violation.

1.1 Our Results

Our first result is an $(O(\log k \log n), O(\log^2 n))$-bicriteria approximation for DB-DST that runs in quasi-polynomial time: We say that a randomized algorithm is an (α, β)-bicriteria-approximation algorithm if it outputs a tree T containing an $r \to t$ path for every terminal $t \in K$ such that the number of children of every vertex v in T is at most $\beta \cdot d_v$, and the expected cost of the tree is at most α times the cost of the optimum tree that does not violate the degree constraints.

▶ **Theorem 1.** *There is a randomized $(O(\log n \log k), O(\log^2 n))$-bicriteria approximation algorithm for the degree-bounded directed Steiner tree problem in $n^{O(\log n)}$-time.*

To the best of our knowledge, our result for DB-DST is the first non-trivial bicriteria approximation for the problem. As we mentioned, the $O(\log n \log k)$-factor for the cost is almost tight due to the hardness results of [15] and [13] for DST. There is a hardness of $\Omega(\log n)$ for the degree violation factor from the Set Cover problem, even if we ignore the cost of the output tree.

▶ Remark. As in [13, 10], we could save a factor of $\log\log n$ in the approximation factor for the problem, with a slight increase in the running time. However, this complicates the algorithmic framework. To deliver the algorithmic idea in a cleaner way, we choose to present the results with $O(\log n \log k)$ approximation ratios.

Our second result is for the degree-bounded group Steiner tree problem on trees (DB-GST-T). We obtain an $\big(O(\log n \log k), O(\log n)\big)$-bicriteria approximation, which is (almost) tight on both factors:

▶ **Theorem 2.** *There is a randomized $\big(O(\log n \log k), O(\log n)\big)$-bicriteria approximation for the degree-bounded group Steiner tree problem on trees.*

1.2 Our Techniques

Our algorithm for DB-DST takes ingredients from both [13] and [10]. As in these papers, we consider an optimum solution, and recursively partition it into balanced sub-trees; we then assign a "state" to each of these sub-trees. The tree structure of this recursive partition, as well as all of the states, form what we call a *state tree*. We solve the problem indirectly, by finding a good state tree, which we can transform back into a corresponding good solution. The state of a sub-tree contains a set of special vertices in the sub-tree that we call *portals*; these were used in [10] to obtain their improved approximation algorithm for DST. We construct a super-tree $\mathbf{T}^\circ$ that contains all possible state trees as sub-trees and reduce the problem considered into that of finding a good sub-tree of small cost in $\mathbf{T}^\circ$. This can be done by formulating a linear program (LP) relaxation and rounding the LP solution using a recursive procedure. The construction of the super-tree and the LP rounding techniques are similar to those in [13]. To extend the algorithm to DB-DST, we need to store the degrees of all of the portals in the state.

This algorithmic framework outputs a so-called "multi-tree": This is a tree where a vertex or an edge can appear multiple times. Repeating the procedure for $Q = O(\log n \log k)$ times, we obtain a set of Q multi-trees. This process violates the degree requirements and thus we obtain bicriteria approximation results. The analysis of this process is non-trivial as we need to prove a concentration bound on the number of times a vertex appears in a multi-tree.

Our technique for DB-GST-T is in observing that the rounding algorithm for GST-T (no degree bounds) in [9] is indeed a generalization of random walk. As we slightly boost the branching probability by a constant factor, this (almost) does not affect the degree bound, but the probability of connecting the root vertex to each group is amplified dramatically. A drawback is that it also incurs a huge blow-up in the cost. To handle the blow-up, we stop amplifying the branching probability when the connecting probability is sufficiently large. The best (but inaccurate) way to illustrate our algorithm is by considering a random walk from the root vertex to a group O_t. We change the random process by branching into two directions simultaneously in each step, and then stop the extra branching when it generates $\Theta(\log n)$ simultaneous random walks. Since we have $O(\log n)$ simultaneous random walks, the cost incurred by the process is blown-up by a factor $O(\log n)$, but the degree-violation is blown-up by only a factor 2. At the same time, the probability of reaching the group O_t goes up by a factor $\Omega(\log n)$. Thus, if we need $O(\log k \log n)$ rounds to reach every group, then we now need only $O(\log k)$ rounds. There is no difference in the cost for running the algorithm for $O(\log k \log n)$ rounds or $O(\log k)$ rounds (with an extra $O(\log n)$ factor in the cost), but it saves a factor in the degree-violation of $O(\log n)$.

2 Preliminaries for Degree-Bounded Directed Steiner Tree

2.1 Notations and Assumptions

In our algorithm and analysis for the DB-DST problem, a tree is always an out-arborescence. Given a tree T, we use $\text{root}(T)$ to denote its root. Given T and a vertex v in T, we use $\Lambda_T(v)$ to denote the set of children of v, and $\Lambda^*_T(v)$ to denote the set of descendants of v

(including v itself) in the tree T. A sub-tree T' of T is a weakly-connected sub-graph of T; such a T' must be an out-arborescence. Sometimes, we shall use left and right children to refer to the two children of a vertex in a tree; in this case, the order of the two children is important and will be clearly specified. For an edge $e = (u, v)$, we use $\text{head}(e) = v$ to denote its head.[1] For a triple $\xi = (u, v, v')$ of three vertices, we use $\text{second}(\xi) = v$ and $\text{third}(\xi) = v'$ to denote the second and third parameter of ξ.

Our input digraph is G. Let $d_{\max} = \max_{v \in V} d_v$. We shall assume each terminal $t \in K$ has only one incoming edge and no outgoing edges in G. This can be assumed w.l.o.g using the following simple operation: For every terminal $t \in K$ that does not satisfy the condition, we add a new vertex t', an edge (t, t') with cost $c_{(t,t')} = 0$ and replace t with t' in K. We increase d_t by 1 and set $d_{t'} = 0$.

One more assumption we can make is that each non-terminal $u \in V \setminus K$ has at most 2 outgoing edges in G. To make sure that this holds, we focus on some non-terminal u with $b \geq 3$ outgoing edges. We replace the star centered at u with its b outgoing edges by a gadget which is a full binary-tree rooted at u with b leaves being the out-neighbors of u. For every newly added vertex u, we set $d_u = d_{\max}$. This way every vertex in G will have at most 2 outgoing edges. The cost of the edges in the gadget can be naturally defined: every leaf edge (of this gadget) ending at w has cost $c_{(u,w)}$, and all internal edges have zero cost. However, this operation changes the degree of vertices. To address this issue, we define a simple transformation function $\phi_v : \mathbb{Z} \to \mathbb{Z}$ for every $v \in V$ as follows: If v is a vertex in the original graph, then ϕ_v is identically 1. Otherwise, v is a non-root internal vertex of some gadget and we define ϕ_v to be the identity function. Then we can compute the *original degree* ρ_u of a vertex u in a tree T of G recursively as follows: $\rho_u = 0$ if u is a leaf, and $\rho_u = \sum_{v \in \Lambda_T(u)} \phi_v(\rho_v)$ otherwise. So, we require that for every v in the output tree T, the original degree ρ_v of v is at most d_v.

2.2 Balanced Tree Partition

We shall use the following basic tool as the starting point of our algorithm design. Its proof is elementary and deferred to Appendix B.

▶ **Lemma 3.** *Let $T = (V_T, E_T)$ be an n-vertex binary tree. Then there exists a vertex $v \in V_T$ with $n/3 < |\Lambda^*_T(v)| \leq 2n/3 + 1$.*

Given a tree $T = (V_T, E_T)$ as in the lemma, we can partition it into two trees $T_1 = (V_{T_1}, E_{T_1})$ and $T_2 = (V_{T_2}, E_{T_2})$, where T_2 contains vertices in $\Lambda^*_T(v)$ and T_1 contains vertices in $V_T \setminus (\Lambda^*_T(v) \setminus \{v\})$. First assume $n \geq 4$. Since $2n/3+1 < n$, we know that $v \neq \text{root}(T)$, thus implying $\text{root}(T_1) = \text{root}(T) \neq \text{root}(T_2) = v$, which is a leaf in T_1. Consequently, we have $E_{T_1} \uplus E_{T_2} = E_T$ and $V_{T_1} \cup V_{T_2} = V_T, V_{T_1} \cap V_{T_2} = \{\text{root}(T_2)\}$. Moreover, $|V_{T_1}|, |V_{T_2}| \leq 2n/3+1$, which is strictly less than n. Thus, T_1 and T_2 are sub-trees that form a balanced partition of (the edges of) T. We call this procedure the *balanced tree partitioning* on T.

When $n = 3$, there are 2 types of trees. If the root has two children, then we could not make both $|V_{T_1}|$ and $|V_{T_2}|$ to be smaller than 3. If the tree is a path of 2 edges, then we can choose v to be the middle vertex and the procedure partitions the tree into two edges. Later, we shall apply the balanced tree partitioning procedure recursively. We stop the recursion when the tree is either an edge, or only contains the root and its 2 children. In other words, the tree has only 1 level of edges.

[1] [12] mistakenly called u the head and v the tail. We follow the common convention to call v the head.

2.3 Multi-Tree

We define a *multi-tree* in G as an intermediate structure. It is simply a tree over *multi-sets* of vertices and edges in G:

▶ **Definition 4** (Multi-Tree). *Given the input digraph $G = (V, E)$, a* multi-tree *in G is a tree $T = (V_T, E_T)$ where every vertex $a \in V_T$ is associated with a label* $\text{label}(a) \in V$ *such that for every $(a, b) \in E_T$, we have* $(\text{label}(a), \text{label}(b)) \in E$.

We say that each vertex $a \in V_T$ is a copy of the vertex $\text{label}(a) \in V$ and each edge $(a, b) \in E_T$ is a copy of the edge $(\text{label}(a), \text{label}(b)) \in E$. So, we say that T is rooted at a copy of $v \in V$, if $\text{label}(\text{root}(T)) = v$, and T contains a copy of some $v \in V$ if there exists some $a \in V_T$ with $\text{label}(a) = v$. We extend the costs c_e, the functions ϕ_v and the degree bounds d_v to their copies in a multi-tree as follows: for a vertex a and an edge (a, b) in a multi-tree, $d_a = d_{\text{label}(a)}, \phi_a \equiv \phi_{\text{label}(a)}$ and $c_{(a,b)} = c_{(\text{label}(a),\text{label}(b))}$. The cost of a multi-tree $T = (V_T, E_T)$ is naturally defined as $\text{cost}(T) = \sum_{e \in E_T} c_e$. Given a multi-tree T, the "original degree" ρ_a of a vertex a can be computed in the same way as before.

▶ **Definition 5** (Good Multi-Trees). *Let $T = (V_T, E_T)$ be a multi-tree in G. We say that T is good if it is rooted at a copy of r, has leaves being copies of terminals, and the original degree of any vertex a in T is at most d_a.*

We can then state the main theorem for DB-DST, which we prove in Sections 3 to 5.

▶ **Theorem 6** (Main Theorem for DB-DST). *There is an $n^{O(\log n)}$-time randomized algorithm that outputs a good multi-tree $T = (V_T, E_T)$ such that*

(6a) $\mathbb{E}_T[\text{cost}(T)] \le \text{opt}$*, where* opt *is the cost of the optimum solution for the instance.*

(6b) *For every $t \in K$, we have* $\Pr_T[V_T \text{ contains a copy of } t] \ge \Omega(1/\log n)$.

(6c) *For some $s = \Omega\left(\frac{1}{\log n}\right)$, it holds, for every $v \in V$, that*

$$\mathbb{E}\left[\exp\left(s \cdot (\textit{number of copies of } v \textit{ in } T)\right)\right] \le 1 + O\left(\tfrac{1}{\log n}\right).$$

We show that this implies Theorem 1.

Proof of Theorem 1. We run the algorithm in Theorem 6 Q times to obtain Q good multi-trees $T_1, T_2, \ldots, T_Q$, for some large enough $Q = O(\log n \log k)$. Our output will contain all edges that appear in the Q multi-trees. Notice that the output may not be a tree, but we can remove edges so that it becomes a tree. Applying union bound, all terminals appear in the union of the Q trees with probability at least 0.9, when Q is big enough. By Property (6c) in the theorem statement, we have for every v,

$$\mathbb{E}\left[\exp\left(s \cdot (\#\text{ copies of } v \text{ in } T_1, \ldots, T_Q)\right)\right] \le \left(1 + O\left(\frac{1}{\log n}\right)\right)^Q = \exp(O(\log k)).$$

The above inequality holds since the Q trees are produced independently.

Thus, if $M = O(\log n)$ is big enough, by Markov's inequality we have

$$\Pr\left[\exp\left(s \cdot (\#\text{ copies of } v \text{ in } T_1, \ldots, T_Q)\right) \ge \exp(M)\right] \le \frac{1}{10n}.$$

The event on the left side is exactly that the number of copies of v in $T_1, \ldots, T_Q$ is at least M/s.

Thus, with probability at least 0.8, every terminal t appears in one of the Q trees and every vertex v appears at most $M/s = O(\log^2 n)$ times in the trees. Taking the union of all trees and reflecting the edges in the original graph G, we have a sub-graph G' of G containing a path

from r to every terminal $t \in K$. The total cost of edges in G' is at most $O(\log n \log k) \cdot \text{opt}$. For every vertex v, the out-degree of v in G' is at most $(M/s)d_v = O(\log^2 n)d_v$. We can take an arbitrary Steiner tree T in G' as the output of the algorithm. This gives us an $(O(\log n \log k), O(\log^2 n))$-bicriteria approximation algorithm for the degree-bounded directed Steiner tree problem. The running time of the algorithm is $n^{O(\log n)}$. ◀

Organization. The remaining part of the paper is organized as follows. In Section 3, we define states and good state trees. In Section 4, we argue that the problem of finding a small cost valid tree can be reduced to that of finding a small cost state-tree. In Section 5, we present our linear programming rounding algorithm that finishes the proof of Theorem 6. Section 6 is dedicated to the proof of Theorem 2 for the degree-bounded group Steiner tree problem on trees (DB-GST-T).

3 States and State-Trees

Given the optimum tree T^* (which is binary by our assumptions) for the DB-DST problem, we can apply the balanced tree partitioning recursively to obtain a *decomposition tree*: We start from T^* and partition it into two trees T_1 and T_2 using the balanced-tree-partitioning procedure, and then recursively partition T_1 and T_2 until we obtain sub-trees with 1 level of edges: Such a tree contains either a single edge, or two edges from the root. Then the decomposition tree is a full binary tree where each node corresponds to a sub-tree of T^*. Due to the balance condition, the height of the tree will be $O(\log n)$. Throughout the paper, we shall use $h = \Theta(\log n)$ to denote an upper bound on the height of this decomposition tree.

Thanks to its small depth, the decomposition tree becomes the object of interest. However, as each node in the tree corresponds to a sub-tree of the optimum solution T^*, it contains too much information for the algorithm to handle. Instead, we shall only extract a small piece of information from each node that we call the *state* of the node. On one hand, a state contains much less information than a sub-tree does, so we can afford to enumerate all possible states for a node. On the other hand, the states of nodes in the decomposition tree still contain enough information for us to check whether the correspondent multi-tree is good. We call the binary tree of states a *state tree*; we require in a *good state tree*, the states of nodes satisfy some consistency constraints. Then we can establish a two-direction connection between good multi-trees and good state trees.

Given a valid tree T in G and a sub-tree T' of T, we now start to make definitions related to the state of T' w.r.t T. It is convenient to think that T is the optimum tree T^* and T' is a sub-tree of $T = T^*$ obtained from the recursive balanced-partitioning procedure since this is how we use the definitions. However, the definitions are w.r.t. general T and T'; from now on till the end of Section 3, we fix any valid tree T and its sub-tree T'.

3.1 Portals

Other than root(T'), the state for T' w.r.t. T contains the set of *portals* of T':

▶ **Definition 7.** *A vertex v in T' is a* portal *in T' if v is* root(T') *or a non-terminal leaf of T'.*

In general, the set of portals of T' can be large, but if T' is obtained from the recursive balanced-tree-partitioning procedure for T, then the number of portals can be shown to be at most $h + 1$. As we shall often use the root and set of portals together, we make the following definition:

▶ **Definition 8** (Root-Portals-Pair). *A pair (r', S) is called a root-portals-pair for vertex r' and portals S if $r' \in S \subseteq V \setminus K$.*

It is easy to see that the root-portal-pairs for an internal node of the decomposition tree and its two children satisfy some properties stated in the following definition:

▶ **Definition 9** (Allowable Child-Pair). *Given three root-portals-pairs $(r', S), (r', S_1)$ and (r'', S_2), we say $((r', S_1), (r'', S_2))$ is an allowable child-pair of (r', S) if $r'' \notin S, S_1 \cup S_2 = S \cup \{r''\}$ and $S_1 \cap S_2 = \{r''\}$.*

The following claim motivates the definition of allowable child pairs:

▷ Claim 10. Assume $T' = (V', E')$ contains at least 2 levels of edges. Let $T'_1 = (V'_1, E'_1)$ and $T'_2 = (V'_2, E'_2)$ be the two sub-trees obtained by applying the balanced tree partitioning on T'. Let $r' = \text{root}(T') = \text{root}(T'_1)$, $r'' = \text{root}(T'_2) \neq r'$ and S, S_1, S_2 be the sets of portals in T', T'_1, T'_2 respectively. Then, $((r', S_1), (r'', S_2))$ is an allowable child-pair of (r', S).

Proof. First, r'' is not a portal of T' since it is a non-root internal vertex in of T'. Second, it is easy to see that $S_1 = (S \cup \{r''\}) \cap V'_1$ and $S_2 = (S \cup \{r''\}) \cap V'_2$. So, $S_1 \cup S_2 = S \cup \{r''\}$ and $S_1 \cap S_2 = \{r''\}$. ◁

3.2 Degree Vectors

The next piece of the information in a state is a *degree vector*:

▶ **Definition 11.** *A degree vector for a set $S \subseteq V \setminus K$ is a vector $\rho = (\rho_v)_{v \in S}$, where ρ_v is an integer in $[1, d_v]$ for every $v \in S$.*

As explained in section 2.1, ρ_v will be the original degree of v in the tree T.

▶ **Definition 12** (Consistency of degree vectors). *Given a root-portals-pair (r', S), an allowable child-pair $((r', S_1), (r'', S_2))$ of (r', S), three degree vectors ρ, ρ^1 and ρ^2 for S, S_1 and S_2 respectively, we say ρ^1 and ρ^2 are consistent with ρ, if*

- *for every $v \in S_1 \setminus \{r''\}$, we have $\rho_v = \rho^1_v$,*
- *for every $v \in S_2 \setminus \{r''\}$, we have $\rho_v = \rho^2_v$ and*
- *$\rho^1_{r''} = \rho^2_{r''}$.*

So, the degree vectors are consistent if there is no contradictory information among them.

▶ **Definition 13** (Edge/Triple Agreeing with Degree Vector). *Given a root-portals-pair (r', S) with $|S| \leq 2$, a degree vector ρ for S, and an edge $(r', v) \in E$ with $\{r', v\} \setminus K = S$, we say (r', v) agrees with ρ if $\rho_{r'} = (\phi_v(\rho_v)$ or $1)$, where $(\phi_v(\rho_v)$ or $1)$ denotes $\phi_v(\rho_v)$ if ρ_v is defined (i.e., if $v \in S$) and 1 otherwise.*

Similarly, given a root-portals-pair (r', S) with $|S| \leq 3$, a degree vector ρ for S, and two edges $(r', v), (r', v') \in E$ such that $\{r', v, v'\} \setminus K = S$, we say the triple (r', v, v') agrees with ρ if $\rho_{r'} = (\phi_v(\rho_v)$ or $1) + (\phi_{v'}(\rho_{v'})$ or $1)$.

Notice that, in the above definition, either $v \in S$ or $v \in K$. In the former case, ρ_v is defined; in the latter case ρ_v is not defined but we know ϕ_v is identically 1. The same argument holds for v'. The definition corresponds to the case when T' is a base case of the recursive balanced tree partitioning, i.e., T' contains only 1 level of edges. If T' contains an edge $e = (r', v)$, then the portal set of T' is $\{r', v\} \setminus K$. We shall have $\rho_{r'} = (\phi_v(\rho_v)$ or $1)$. Similarly, if T' contains 3 vertices (r', v, v') with r' being the root, then we must have $\rho_{r'} = (\phi_v(\rho_v)$ or $1) + (\phi_{v'}(\rho_{v'})$ or $1)$.

3.3 States and Good State-Trees

With degree vectors, we can define states and good state-trees:

▶ **Definition 14.** *A* state *is a tuple (r', S, ρ) where (r', S) is a root-portals-pair and ρ is a degree vector for S.*

The state of the tree T' w.r.t T is the tuple (r', S, ρ) with $r' = \mathrm{root}(T')$, S being the set of portals in T', and ρ being the vector of original degrees of vertices in S w.r.t. the tree T.

▶ **Definition 15** (Good State Trees)**.** A good state tree *is a full binary tree τ of depth at most h, where every node p is associated with a state (r'_p, S_p, ρ^p), and every leaf o is associated with either an edge $e_o \in E$ or a triple ξ_o such that the following conditions hold.*

(15a) $\left(r'_{\mathrm{root}(\tau)}, S_{\mathrm{root}(\tau)}\right) = (r, \{r\})$.

(15b) *For any leaf o of τ, either e_o or ξ_o agrees with ρ^o.*

(15c) *For an internal node p in τ, letting q and o be the left and right children of p, then the pair $((r'_q, S_q), (r'_o, S_o))$ is an allowable child-pair of (r'_p, S_p) (so, $r'_q = r'_p \neq r'_o$), and ρ^q and ρ^o are consistent with ρ^p.*

We say that a terminal $t \in K$ is involved *in a good state tree τ if there exists a leaf o of τ with $t = \mathrm{head}(e_o)$, or $t \in \{\mathrm{second}(\xi_o), \mathrm{third}(\xi_o)\}$.*

Given a good state tree τ, and a leaf o in τ, we define the cost $c(o)$ as follows. If e_o is defined, then we define $c(o) = c_{e_o}$; otherwise, define $c(o) = c_{(r'_o, \mathrm{second}(\xi_o))} + c_{(r'_o, \mathrm{third}(\xi_o))}$. The cost of a state-tree τ is defined as $\mathrm{cost}(\tau) := \sum_{o \text{ leaf of } \tau} c(o)$.

4 Reduction to Finding Good State-Trees

4.1 From a Valid Tree to a Good State-Tree Involving All Terminals

In this section, we show that the decomposition tree of the optimum tree T^* can be turned into a good state tree τ^* with desired properties. The procedure in the proof of the following lemma is only for the analysis purpose and is not a part of our algorithm.

▶ **Lemma 16.** *There is a good state-tree τ^* involving all terminals with $\mathrm{cost}(\tau^*) = \mathrm{cost}(T^*)$.*

Proof. As we alluded, the state tree τ^* is constructed by taking the state for each node in the decomposition tree for T^*. Formally, it is obtained by calling gen-state-tree(T^*) (defined in Algorithm 1). In the algorithm ρ^{T^*} is the vector of original degrees of all vertices in T^*.

Algorithm 1 gen-state-tree(T').

create a node p with $r'_p = \mathrm{root}(T'), S_p =$ portals of T' and ρ^p being ρ^{T^*} restricted to S_p
if T' has only 1 level of edges **then**
    if T' contains a single edge e **then** let $e_p = e$ and **return** the single node p
    otherwise, T' contains two edges (r', v) and (r', v'), let $\xi_p = (r', v, v')$ and **return** p
apply balanced tree partitioning to decompose T' into T'_1 and T'_2
$\tau_1 \leftarrow$ gen-state-tree(T'_1), $\tau_2 \leftarrow$ gen-state-tree(T'_2)
return the tree τ obtained by combining p, τ_1 and τ_2 with edges $(p, \mathrm{root}(\tau_1))$ and $(p, \mathrm{root}(\tau_2))$, with $\mathrm{root}(\tau_1)$ and $\mathrm{root}(\tau_2)$ being the left and right children of p respectively

We first show that τ^* is a good state tree, by showing that it satisfies all the properties in Definition 15. Property (15a) trivially holds by the way we define the parameters for the root recursion of gen-state-tree. Property (15b) holds by that each ρ^p is ρ^{T^*} restricted to S^p. Property (15c) follows from the same facts and Claim 10. $\text{cost}(\tau^*) = \sum_{e \in E_{T^*}} c_e = \text{cost}(T^*)$ since every edge in T^* is counted exactly once in τ^*. ◀

4.2 From a Good State Tree to a Good Multi-Tree

Now we focus on the other direction of the reduction and prove the following lemma:

▶ **Lemma 17.** *Given a good state tree τ, we can efficiently construct a good multi-tree T with $\text{cost}(T) = \text{cost}(\tau)$. Moreover, if a terminal $t \in K$ is involved in τ, then T contains a copy of t.*

Proof. The multi-tree T is constructed by joining the edges associated with all leaf nodes o in τ using a recursive procedure. For each node p in τ, we shall construct a multi-tree T_p for p, as well as a mapping π_p from S_p to vertices in T_p. The multi-tree T_p and the mapping π_p satisfy the following properties:

(P1) For every $v \in S_p$, we have $\text{label}(\pi_p(v)) = v$; that is, $\pi_p(v)$ is a copy of v.

(P2) $\pi_p(r'_p) = \text{root}(T_p)$.

In particular, the two properties imply that $\text{root}(T_p)$ is a copy of r'_p.

The trees and mappings are constructed from the bottom to the top of the tree τ. Focus on a leaf node p with $e_p = (r', v)$. If e_p is defined, then T_p only contains a copy of the edge (r', v). π_p maps r' to the copy of r', and if $v \notin K$ (thus, $v \in S_p$), v to the copy of v in T_p. Otherwise ξ_p is defined. Then T_p contains a tree with two edges: a copy of $(r'_p, \text{second}(\xi_p))$ and a copy of $(r'_p, \text{third}(\xi_p))$. π_p can also be defined naturally.

Now consider the case that p is an internal node and let q and o be its left and right children. Then, we have $r'_p = r'_q, r'_o \notin S_p, S_q \cup S_o = S_p \cup \{r'_o\}$ and $S_q \cap S_o = \{r'_o\}$ by Property (15c). Then we identify $\pi_q(r'_o)$ with $\pi_o(r'_o) = \text{root}(T_o)$, and then the multi-tree T_p is the new tree containing vertices in T_q and T_o. Notice that both $\pi_q(r'_o)$ and $\pi_o(r'_o)$ are copies of r'_o; thus, the obtained T_p can be well-defined. The mapping π_p is just the combination of π_q and π_o: For a vertex $v \in S_q$, let $\pi_p(v) = \pi_q(v)$; for a vertex $v \in S_o$, let $\pi_p(v) = \pi_o(v)$; since $S_q \cap S_o = \{r'_o\}$ and we identified $\pi_q(r'_o)$ with $\pi_o(r'_o)$, the mapping is well-defined. Also, it is easy to see that (P1) and (P2) holds for T_p and π_p.

Our final multi-tree for τ will be $T = T_{\text{root}(\tau)}$. It is straightforward to see that if $t \in K$ is involved in τ, then T contains a copy of t. Notice that all the ρ^p-vectors are consistent with each other, and for every leaf o, e_o or ϵ_o agrees with ρ^o. Thus, aggregating all the ρ^p vectors will recover the vector ρ^T of original degrees of vertices in ρ^T. So, the multi-tree T is good since every v in T has $\rho^T_v \in [1, d_v]$. The cost of T is $\sum_{e \in E_T} c_e = \sum_{o:\text{ leaves of } \tau} c(o) = \text{cost}(\tau)$. Finally, the running time of the algorithm is polynomial in the size of the state-tree τ. ◀

5 Finding a Good State Tree using LP Rounding

5.1 Extended State Trees and Construction of $\mathbf{T^0}$

With the relationship between good multi-trees and good state trees established, we can now focus on the problem of finding a good state-tree of small cost involving many terminals. We shall construct a quasi-polynomial sized tree $\mathbf{T}^\circ$ so that every good state-tree τ corresponds to a sub-tree $\mathbf{T}$ of $\mathbf{T}^\circ$ satisfying some property. Roughly speaking, $\mathbf{T}^\circ$ is the "super-set" of all potential good state-trees τ. However, since the consistency conditions are defined over three states for a parent and its two children, it is more convenient to insert a "virtual" node

between every internal node and its two children. Also, it is convenient to break a leaf state node o into two nodes, one containing the state information and the other containing e_o or ξ_o. Formally, for a good state-tree τ, we construct a correspondent tree $\mathbf{T}$ as follows.

1. Let $\mathbf{T}$ be a copy of τ. All nodes in $\mathbf{T}$ are called *state nodes*.
2. For every internal state node p in $\mathbf{T}$ with left and right children p_1 and p_2, we create a *virtual node* q and replace the two edges (p, p_1) and (p, p_2) with 3 edges $(p, q), (q, p_1)$ and (q, p_2); p_1 is still the left child and p_2 is the right child.
3. For every leaf state node p, we create a *base node* o and let o be the child of p. Then we move the e_p or ξ_p information from the node p to node o: If e_p is defined, then we let $e_o = e_p$ and undefine e_p; otherwise, let $\xi_o = \xi_p$ and undefine ξ_p.
4. We add a *super node* $\mathbf{r}$ and an edge from $\mathbf{r}$ to the root of $\mathbf{T}$. $\mathbf{r}$ will be the new root for $\mathbf{T}$.

We call this $\mathbf{T}$ the *extended state-tree* for τ; we say $\mathbf{T}$ is good if its correspondent τ is good. Clearly, there is a 1-to-1 correspondence between good state trees and good extended state trees.

Our $\mathbf{T}^\circ$ will be the "super-set" of all potential good extended state trees $\mathbf{T}$. Formally, we create a super node $\mathbf{r}$ to be the root of $\mathbf{T}^\circ$. Then, for every $\rho_r \in [1, d_r]$, we call cnstr-$\mathbf{T}^\circ(0, r, \{r\}, \rho = (\rho_r))$ to obtain a tree and let its root be a child of $\mathbf{r}$.

Algorithm 2 cnstr-$\mathbf{T}^\circ(h', r', S, \rho)$.

create a state node p with $(r'_p, S_p, \rho^p) = (r', S, \rho)$
for every $(r', v) \in E$ such that $\{r', v\} \setminus K = S$ and (r', v) agrees with ρ **do**
    create a "base node" o with $e_o = (r', v)$ and let o be a child of p
    let $c(o) = c_{(r',v)}$
for every $(r', v), (r', v') \in E$ such that $\{r', v, v'\} \setminus K = S$ and (r', v, v') agrees with ρ **do**
    create a "base node" o with $\xi_o = (r', v, v')$ and let o be a child of p
    let $c(o) = c_{(r',v)} + c_{(r',v')}$
if $h' < h$ **then**
    for every allowable child-pair $((r', S_1), (r'', S_2))$ of (r', S) **do**
        for every pair of degree vectors ρ^1 for S_1 and ρ^2 for S_2 such that ρ^1 and ρ^2 are consistent with ρ **do**
            create a "virtual node" q and let q be a child of p
            $\mathbf{T}_1 \leftarrow$ cnstr-$\mathbf{T}^\circ(h' + 1, r', S_1, \rho^1)$
            $\mathbf{T}_2 \leftarrow$ cnstr-$\mathbf{T}^\circ(h' + 1, r'', S_2, \rho^2)$
            let the left and right sub-trees of q be $\mathbf{T}_1$ and $\mathbf{T}_2$ respectively
return the tree $\mathbf{T}$ rooted at p

The following claim is immediate from the construction of $\mathbf{T}^\circ$.

▷ Claim 18. A subtree $\mathbf{T}$ of $\mathbf{T}^\circ$ with root($\mathbf{T}$) = root($\mathbf{T}^\circ$) is a good extended state tree if and only if the following happens:

- The super node in $\mathbf{T}$ has exactly one child (which is a state node).
- Each state node in $\mathbf{T}$ has exactly one child (which is a base node or a virtual node).
- For each virtual node q in $\mathbf{T}$, both q's children in $\mathbf{T}^\circ$ are in $\mathbf{T}$.

On the other hand, every good extended tree $\mathbf{T}$ of depth at most $h + 1$ is a sub-tree of $\mathbf{T}^\circ$ with root being root($\mathbf{T}^\circ$).

Also, we say that a vertex v is involved in $\mathbf{T}$ if there is a base node o in $\mathbf{T}$ with $v = \text{head}(e_o)$ or $v \in \{\text{second}(\xi_o), \text{third}(\xi_o)\}$. The cost of $\mathbf{T}$, denoted as cost($\mathbf{T}$), is defined as the sum of $c(o)$ over all base nodes in $\mathbf{T}$. So, the problem now becomes finding a small-cost good extended state tree in $\mathbf{T}^\circ$ that involves each terminal with large probability.

5.2 LP Formulation

We formulate an LP relaxation for our task. Let $\mathbf{V}^\circ$ be the set of nodes in $\mathbf{T}^\circ$, $\mathbf{r} = \text{root}(\mathbf{T}^\circ)$ and let $\mathbf{V}^\circ_{\text{state}}, \mathbf{V}^\circ_{\text{virt}}$ and $\mathbf{V}^\circ_{\text{base}}$ be the sets of state, virtual and base nodes in $\mathbf{T}^\circ$ respectively. Notice that there is only one super node, which is the root $\mathbf{r}$. For every $v \in V$, let $\mathbf{O}_v = \{v \in \mathbf{V}^\circ_{\text{base}} : v = \text{head}(e_o) \text{ or } v \in \{\text{second}(\xi_o), \text{third}(\xi_o)\}\}$ be the set of base nodes involving v. Let $\mathbf{T}^*$ be our target good extended state tree; this is the tree correspondent to the good state tree τ^*. Then, in our LP, we have a variable x_p for every $p \in \mathbf{V}^\circ$, that indicates whether p is in the $\mathbf{T}^*$ or not.

$$\min \quad \sum_{o \in \mathbf{V}^\circ_{\text{base}}} x_o c(o) \tag{1}$$

$$\sum_{q \in \Lambda_{\mathbf{T}^\circ}(p)} x_q = x_p, \quad \forall p \in \mathbf{V}^\circ_{\text{state}} \cup \{\mathbf{r}\} \tag{2}$$

$$x_p = x_q, \quad \forall q \in \mathbf{V}^\circ_{\text{virt}}, p \in \Lambda_{\mathbf{T}^\circ}(q) \tag{3}$$

$$x_p \in [0,1], \forall p \in \mathbf{V}^\circ \tag{4}$$

$$\sum_{o \in \Lambda^*_{\mathbf{T}^\circ}(p) \cap \mathbf{O}_t} x_o \le x_p, \quad \forall p \in \mathbf{V}^\circ, t \in K \tag{5}$$

$$\sum_{o \in \mathbf{O}_t} x_o = 1, \quad \forall t \in K \tag{6}$$

The objective function of LP (1) is to minimize the total cost of all leaves in $\mathbf{T}^*$. Constraint (2) requires that for every state or super node p in $\mathbf{T}^*$, exactly one child of p is in $\mathbf{T}^*$. Constraint (3) requires that a virtual node q in $\mathbf{T}^*$ has both its children in $\mathbf{T}^*$. Constraint (5) says for every node p in $\mathbf{T}^*$ and every terminal $t \in K$, there is a most one descendant base node o of p that is in $\mathbf{O}_t$. In the whole tree $\mathbf{T}^*$, exactly one leaf node o has $t = \text{head}(e_o)$ or $t \in \{\text{second}(\xi_o), \text{third}(\xi_o)\}$, for every $t \in K$ (Constraint (6)); in the LP, all the variables are between 0 and 1 (Constraint (4)).

Notice that (5) for $p = \mathbf{r}$ and any $t \in K$ and (6) for the same t imply that $x_{\mathbf{r}} = 1$. Constraint (2) and (3) imply that the x values over the nodes of a root-to-leaf path in $\mathbf{T}^\circ$ are non-increasing.

5.3 Rounding Algorithm

Given a valid solution x to LP (1), our rounding algorithm will round it to obtain the set $\mathbf{V} \subseteq \mathbf{V}^\circ$, which induces a good state tree. The algorithm is very similar to that of [9] with the only one difference: For every state node or super-node p that is added to $\mathbf{V}$, we add exactly one child q of p to $\mathbf{V}$, while the algorithm of [9] makes independent decisions for each child. The algorithm is formally described in Algorithm 3. In the main algorithm, we simply call round($\mathbf{r}$). It is straightforward to see that the tree induced by round($\mathbf{r}$) is a good extended state tree. The following claim also holds:

▷ Claim 19. Let $p \in \mathbf{V}^\circ$ and $q \in \Lambda^*_{\mathbf{T}^\circ}(p)$. Let $\mathbf{V}$ be the random set returned by round(p). Then we have $\Pr[q \in \mathbf{V}] = \frac{x_q}{x_p}$.

Applying the above claim for $p = \mathbf{r}$ and every $q \in \mathbf{V}^\circ_{\text{base}}$, we have that the expected cost of the tree induced by $\mathbf{V}$ is exactly $\text{cost}(x)$.

The main theorem we need about the rounding algorithm is as follows:

Algorithm 3 round(p).

1: **if** $p \in \mathbf{V}^{\circ}_{\text{state}} \cup \{\mathbf{r}\}$ **then**
2: randomly choose a child q of p according to probability vector $\left(\frac{x_q}{x_p}\right)_{q \in \Lambda_{\mathbf{T}^\circ}(p)}$
3: **return** $\{p\} \cup \text{round}(q)$
4: **else if** $p \in \mathbf{V}^{\circ}_{\text{virt}}$ **then**
5: **return** $\{p\} \cup \text{round}(\text{left child of } p) \cup \text{round}(\text{right child of } p)$
6: **else return** $\{p\}$

▶ **Theorem 20.** *Let* $\mathbf{V}$ *be the random set returned by* round($\mathbf{r}$). *Then, for any terminal* $t \in K$ *we have* $\Pr[\mathbf{V} \cap \mathbf{O}'_t \neq \emptyset] \geq \frac{1}{h+1}$.

Theorem 20 was proved [9] for the original rounding algorithm and was reproved in [23]. However, adapting the analysis to our slightly different rounding algorithm is straightforward and thus we omit the proof of the theorem here.

We now wrap up and finish the proof of the main theorem (Theorem 6) except for Property (6c), of which the proof is deferred to Appendix A due to the space constraint. We solve LP(1) to obtain a solution x. Notice that $\text{cost}(x) \leq \text{cost}(\mathbf{T}^*) = \text{cost}(\tau^*) = \text{cost}(T^*)$. Let $\mathbf{V} \leftarrow \text{round}(\mathbf{r})$. Then by Claim 18 and the rounding algorithm, the tree $\mathbf{T}$ induced by $\mathbf{V}$ is a good extended state tree. Let τ be the good state tree correspondent to $\mathbf{T}$, and let T be the good multi-tree in G constructed using the procedure in Section 4.2. The cost of the multi-tree T is at most $\text{cost}(x)$. By Theorem 20, for every $t \in K$, the probability that t is involved T is at least $1/(h+1) = \Omega(1/\log n)$.

Let us consider the running time of the algorithmic framework, which is polynomial on the size of the tree $\mathbf{T}^\circ$. First notice that if $((r', S_1), (r'', S_2))$ is an allowable child pair of (r', S), then we have $|S_1|, |S_2| \leq |S| + 1$ since $S_1 \cup S_2 = S \cup \{r''\}$. Thus, a state-node p at the h'-th level in $\mathbf{T}^\circ$ (the children of $\mathbf{r}$ have level 0 and for simplicity we do not consider super and virtual nodes when counting levels) has $|S_p| \leq h' + 1$. Thus, every state node p in $\mathbf{T}^\circ$ has $|S_p| \leq h + 1$.

Then we consider the degree of the tree $\mathbf{T}^\circ$, which is the maximum number of possible children of a state node p with $(r'_p, S_p, \rho^p) = (r', S, \rho)$. First, there are at most $n \times 2^{|S_p|} \leq n \cdot 2^{h+1}$ different allowable child pairs $((r', S_1), (r'', S_2))$ of the pair (r', S): there are at most n choices for r'' and 2^h ways to split S into S_1 and S_2. Then, for a fixed allowable child pair $((r', S_1), (r'', S_2))$, we consider the number of pairs of degree vectors $\left(\rho^1, \rho^2\right)$ such that ρ^1 and ρ^2 are consistent with ρ. This is determined by the value of $\rho^1_{r''} = \rho^2_{r''}$, which has at most $d_{\max}$ possibilities. So, the number of virtual children of a state node is at most $n \cdot 2^{h+1} \cdot d_{\max} = \text{poly}(n)$ since $h = O(\log n)$. The number of child base nodes of p is at most n^2. Since the height of the tree $\mathbf{T}^\circ$ is at most $O(\log n)$, its size is bounded by $(\text{poly}(n))^{O(\log n)} = n^{O(\log n)}$. So the running time of the LP rounding algorithm is $n^{O(\log n)}$. This finishes the proof of Theorems 6 except for Property (6c), whose is deferred to Appendix A.

6 Bicriteria-Approximation Algorithm for Degree-Bounded Group Steiner Tree on Trees

In this section, we prove Theorem 2, which is repeated here.

▶ **Theorem 2.** *There is a randomized* $\left(O(\log n \log k), O(\log n)\right)$*-bicriteria approximation for the degree-bounded group Steiner tree problem on trees.*

We first set up some notations for the theorem. Recall that T° is the input tree, V° denotes the set of vertices of T°, and r denotes the root of T°. For simplicity, we assume the costs are on the vertices instead of edges: Every vertex $u \in V^\circ$ has a cost $c_u \geq 0$. Notice that this does not change the problem. We have k groups indexed by $[k]$. For each group $t \in [k]$, we are given a set $O_t \subseteq V^\circ$ of leaves in T°. W.l.o.g, we assume all O_t's are disjoint. Every vertex $v \in V$ is given a degree bound D_v. The goal of the problem is then to output the smallest cost subtree T of T° that satisfies the degree constraints and contains the root r and one vertex from each O_t, $t \in [k]$. Since now we only have one tree T°, we use the following notations for children and descendants: For every vertex $u \in V^\circ$, let Λ_u denote the set of children of u in T°, and Λ_u^* to denote the set of descendants of u in T° (including u itself).

Now we describe the LP relaxation we use for our problem. For every vertex $u \in T^\circ$, we use x_u to indicate whether u is chosen or not (in the correspondent integer program). LP (7) is a valid LP relaxation for the DB-GST-T problem:

$$\min \quad \sum_{u \in V^\circ} c_u x_u \qquad \text{s.t.} \tag{7}$$

$$x_v \leq x_u \qquad \forall u \in V^\circ, v \in \Lambda_u \tag{8}$$

$$\sum_{o \in O_t} x_o = 1 \qquad \forall t \in [k] \tag{9}$$

$$\sum_{o \in O_t \cap \Lambda_u^*} x_o \leq x_u \qquad \forall t \in [k], \forall u \in V^\circ \tag{10}$$

$$\sum_{v \in \Lambda_u} x_v \leq d_u \cdot x_u \qquad \forall u \in V^\circ \tag{11}$$

$$x_u \in [0,1] \qquad \forall u \in V^\circ \tag{12}$$

In the correspondent integer program, the objective we try to minimize is $\sum_{u \in V^\circ} c_u x_u$, i.e, the total cost of all verticies we choose. Constraint (8) says that if we choose a vertex v, then we must choose its parent u. Constraint (9) requires for every group t, exactly one vertex in O_t is added to the tree. Constraint (10) holds since if u is chosen, at most one vertex in $\Lambda_u^* \cap O_t$ is chosen for every group t. Constraint (11) is the degree constraint. In the LP relaxation, we require each x_u to take value in $[0,1]$ (Constraint (12)). Notice that (9) and (10) for the root r imply that $x_r = 1$.

Modifying the LP solutions. Solving LP (7), we can obtain an optimum LP solution $(x_u)_{u \in V^\circ}$. In our rounding algorithm, it would be convenient if every x_u is a (non-positive) integer power of 2 that is not too small. So, we shall modify the LP solution using the following operations, which may violate many of the LP constraints slightly. For every $v \in V^\circ$ with $x_v < \frac{1}{2n}$, we change x_v to 0. This can only decrease the cost of the solution. It is easy to see that Constraints (8), (10) and (11) will not be violated. Constraint (9) may not hold any more, but we still have $\sum_{v \in O_t} x_v \geq 1 - n \times \frac{1}{2n} \geq \frac{1}{2}$ for every $t \in [k]$. We can remove all vertices v with $x_v = 0$ from the instance and thus assume $x_v \geq \frac{1}{2n}$ for every $v \in V^\circ$. Next, we increase each x_v to the smallest (non-positive) integer power of 2 that is greater than or equal to x_v. This will violate many constraints in the LP by a factor of 2. We list the properties that our new vector $(x_u)_{u \in V^\circ}$ has:

(P1) For every $u \in V^\circ$, x_u is an integer power of 2 between $\frac{1}{2n}$ and 1.
(P2) The x values along any root-to-leaf path in T° is non-increasing.
(P3) $\sum_{o \in O_t} x_o \in [\frac{1}{2}, 2]$ for every group $t \in [k]$.

(P4) $\sum_{o\in O_t\cap\Lambda_u^*} x_o \le 2x_u$ for every $t\in[k]$ and $u\in V^\circ$.
(P5) $\sum_{v\in\Lambda_u} x_v \le 2d_u x_u$ for every $u\in V^\circ$.
(P6) $\sum_{u\in V^\circ} c_u x_u \le 2\cdot\mathrm{opt}$, where opt is the cost of the optimum integer solution.

6.1 The rounding algorithm

We now describe our rounding algorithm. We define two important global parameters: $L := \lceil\log(2n)\rceil$ and $\gamma := \lfloor\log L\rfloor - 2$. We say an edge (u,v) with $v\in\Lambda_u$ has "hop value" 1 if $x_u < x_v$ and 0 if $x_u = x_v$. For every vertex $u\in V^\circ$, we define ℓ_u to be the sum of hop values over all edges in the path from the root to u in T°. Thus, for every $u\in V^\circ$ and $v\in\Lambda_u$, we have $\ell_v - \ell_u \in\{0,1\}$, and $\ell_v = \ell_u$ if and only if $x_v = x_u$. By Properties (P1) and (P2), we have that $\ell_v\in[0,L]$ for every $v\in V^\circ$.

Our rounding algorithm is applied on some scaled solution x', which is defined as follows:

$$x'_u = 2^{\min\{\ell_u,\gamma\}}x_u, \text{for every } u\in V^\circ.$$

As we mentioned in the introduction, this change will increase the probability of choosing v conditioned on choosing u by a factor of 2, for some $u\in V^\circ, v\in\Lambda_u$ with $\ell_u < \ell_v \le \gamma$.

We prove one important property for x', which is necessary for us to run the recursive rounding algorithm.

▷ Claim 21. For every $u\in V^\circ$ and $v\in\Lambda_u$, we have $x'_v \le x'_u$.

Proof. If $x_v = x_u$, then the edge (u,v) has hop value 0 and thus $\ell_v = \ell_u$. In this case we have $x'_v = x'_u$ as well. Otherwise, we have $x_v \le x_u/2$ and $h_v = h_u + 1$. So, $\min\{h_v,\gamma\} \le \min\{h_u,\gamma\}+1$ and therefore $x'_v \le x'_u$. ◁

Notice that $x'_r = 1$ and every x'_v is an integer power of 2 between 2^{-L} and 1. Our recursive rounding algorithm is run over x'. In the procedure recursive-rounding(u), we add u to our output tree and do the following: for every $v\in\Lambda_u$, with probability x'_v/x'_u independent of all other choices, we call recursive-rounding(v). In the root recursion, we shall call recursive-rounding(r).

Our final algorithm will repeat the recursive procedure M times independently, for a large enough $M = O(\log k)$. Let $T_1, T_2, \cdots, T_M$ be the M trees we obtained from the M repetitions. Our final tree T will be the union of the M trees.

We first analyze the expected cost of T. First focus on the tree T_1. It is easy to see that the probability u is chosen by T_1 is exactly $x'_u \le 2^\gamma x_u = O(L)x_u$. Therefore, the expected cost of T_1 is at most $O(L)\cdot\mathrm{opt}$ by Property (P6). Therefore, the expected cost of the tree T is at most $O(ML)\cdot\mathrm{opt} = O(L\log k)\cdot\mathrm{opt} = O(\log n\log k)\cdot\mathrm{opt}$.

We then analyze the degree constraints on T. Given that u is selected by T_1, the probability that we select a child of v of u is $\frac{x'_v}{x'_u} \le \frac{2x_v}{x_u}$. By Property (P5), we have $\sum_{v\in\Lambda_u}\frac{x'_v}{x'_u} \le \sum_{v\in\Lambda_u}\frac{2x_v}{x_u} \le 4d_u$. Consider all the M trees $T_1, T_2, \cdots, T_M$. Even if we condition on the event that u appears in all the M trees, the degree of u is the summation of many independent random $\{0,1\}$-variables. The expectation of the summation is at most $4Md_u = O(\log k)\cdot d_u$. Using Chernoff bound, one can show that the probability that the degree of u is more than $O(\log n)\cdot d_u$ is at most $\frac{1}{10n}$, for some large enough $O(\log n)$ factor. Therefore, with probability at least 0.9, every node u in T has degree at most $O(\log n)\cdot d_u$. Therefore, we proved that the degree violation factor of our algorithm is $O(\log n)$, as claimed in Theorem 2.

6.2 Analysis of connectivity probability

It remains to show that with high probability, the tree T contains a vertex from every group. This is the goal of this section. Till the end of the section, we focus on the tree T_1 and a fixed group t. For every vertex $u \in V^\circ$, we define $\mathbf{E}_u$ to be the event that u is chosen by T_1. Our goal is to give a lower bound on $\Pr[\bigvee_{o\in O_t} \mathbf{E}_o]$, i.e., the probability that some vertex in O_t is chosen by the tree T_1.

Notice that when two adjacent nodes in T° have the same x' value, then the child is chosen whenever the parent is. Thus, we can w.l.o.g contract any sub-tree of nodes in T° with the same x' value into one single super-vertex, without changing the rounding algorithm. Notice that if two adjacent vertices $u \in V^\circ, v \in \Lambda_u$ have $\ell_u = \ell_v$, then we have $x_u = x_v$ and thus $x'_u = x'_v$. So, we contract every maximal sub-tree of vertices in T° with the same ℓ value. After this operation, for every $u \in V^\circ$, ℓ_u is exactly the level of u in the tree T°. So, for every $u \in V^\circ$ and $v \in \Lambda_v$ we have $\ell_v = \ell_u + 1$. A super-vertex is in O_t if one of its vertices before contracting is in O_t. If an internal super-vertex is in O_t, we can remove all its descendants without changing the analysis in this section. So, again we have that O_t only contains leaves.

For every vertex u, we define $z_u := \sum_{o\in O_t\cap\Lambda^*_u} x_o$. Notice that $z_u \le 2x_u$ by Property (P4).

In the following, we shall bound $\Pr\left[\bigvee_{o\in O_t\cap\Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right]$ for every $u \in V^\circ$ from bottom to top. This is done in two stages due to the threshold γ we used when we define x' variables. First we consider the case when $\ell_u \ge \gamma$ and then we focus on the case when $\ell_u < \gamma$. The two stages are captured by Lemmas 22 and 23 respectively.

▶ **Lemma 22.** *For a vertex u with $\ell_u \ge \gamma$, we have* $\Pr\left[\bigvee_{o\in O_t\cap\Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right] \ge \frac{1}{2(L+1-\ell_u)} \frac{z_u}{x_u}$.

Similar lemmas have been proved multiple times in many previous results. Since our parameters are slightly different, we provide the complete proof in Appendix B. The lemma implies that for every u with $\ell_u \ge \gamma$, we have $\Pr\left[\bigvee_{o\in O_t\cap\Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right] \ge \frac{1}{2L} \cdot \frac{z_u}{x_u}$.

Now we analyze the probability for u with $\ell_u \le \gamma$. Recall that $\gamma = \lfloor \log L \rfloor - 2$ and thus we have $2^\gamma \in (L/8, L/4]$. Let $\alpha_\gamma = \frac{1}{2L}$ and for every $\ell \in [0, \gamma-1]$, define $\alpha_\ell = 2\alpha_{\ell+1} - 4\alpha^2_{\ell+1}$. It is easy to see that for every $\ell \in [0,\gamma]$, we have $\alpha_\ell \le \frac{2^{\gamma-\ell}}{2L}$. Then, we have for every $\ell \in [0, \gamma-1]$,

$$\begin{aligned}\alpha_\ell &= 2\alpha_{\ell+1} - 4\alpha^2_{\ell+1} = 2\alpha_{\ell+1}(1 - 2\alpha_{\ell+1})\\ &\ge 2\alpha_{\ell+1}\left(1 - 2\times\frac{2^{\gamma-\ell-1}}{2L}\right) = 2\alpha_{\ell+1}\left(1 - \frac{2^{\gamma-\ell-1}}{L}\right).\end{aligned}$$

Therefore, we have

$$\alpha_0 \ge 2^\gamma \prod_{\ell=1}^{\gamma}\left(1 - \frac{2^{\gamma-\ell-1}}{L}\right)\alpha_\gamma \ge \frac{2^\gamma}{2L}\prod_{\ell=1}^{\gamma} e^{-2^{\gamma-\ell}/L} \ge \frac{2^\gamma}{2L} e^{-2^\gamma/L} = \Omega(1).$$

The second inequality used that $1-\theta \ge e^{-2\theta}$ for every $\theta \in (0, 1/2)$. The last equality used that $\gamma = \lfloor \log L \rfloor - 2$ and thus $2^\gamma = \Theta(L)$.

With the α values defined, we also have the following lemma for the case when $\ell_u \le \gamma$:

▶ **Lemma 23.** *For every vertex $\ell_u = \ell \le \gamma$, we have* $\Pr\left[\bigvee_{o\in O_t\cap\Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right] \ge \alpha_\ell \frac{z_u}{x_u}$.

The proof of this lemma is deferred to Appendix B. Applying the lemma for the root r of T°, we have that $\Pr\left[\bigvee_{o\in O_t} \mathbf{E}_o\right] \ge \alpha_0 \cdot \frac{z_r}{x_r} \ge \alpha_0 \cdot \frac{1}{2} = \Omega(1)$.

Now we consider all the M trees $T_1, T_2, \cdots, T_M$ together. The probability that O_t is not chosen by any of the M trees is at most $(1 - \Omega(1))^M \leq \frac{1}{10k}$, if our $M = O(\log k)$ is big enough. Thus, the probability that T, the union of all trees $T_1, T_2, \cdots, T_M$, contains an r-to-O_t path for every t, is at least 0.9.

References

1 Yair Bartal. Probabilistic approximations of metric spaces and its algorithmic applications. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 184–193, 1996. doi:10.1109/SFCS.1996.548477.

2 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33(1):73–91, 1999. doi:10.1006/jagm.1999.1042.

3 Sina Dehghani, Soheil Ehsani, Mohammad Taghi Hajiaghayi, Vahid Liaghat, Harald Räcke, and Saeed Seddighin. Online weighted degree-bounded steiner networks via novel online mixed packing/covering. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 42:1–42:14, 2016. doi:10.4230/LIPIcs.ICALP.2016.42.

4 Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, and Vahid Liaghat. Online degree-bounded steiner network design. In *Proceedings of the Twenty-seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '16, pages 164–175, Philadelphia, PA, USA, 2016. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=2884435.2884448.

5 Sina Dehghani, Soheil Ehsani, MohammadTaghi Hajiaghayi, Vahid Liaghat, and Saeed Seddighin. Greedy algorithms for online survivable network design. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 152:1–152:14, 2018. doi:10.4230/LIPIcs.ICALP.2018.152.

6 Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004. doi:10.1016/j.jcss.2004.04.011.

7 Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, pages 285–296, 2014. doi:10.1007/978-3-319-07557-0_24.

8 Martin Fürer and Balaji Raghavachari. Approximating the minimum-degree steiner tree to within one of optimal. *J. Algorithms*, 17(3):409–423, 1994. doi:10.1006/jagm.1994.1042.

9 Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000. doi:10.1006/jagm.2000.1096.

10 Rohan Ghuge and Viswanath Nagarajan. A quasi-polynomial algorithm for submodular tree orienteering in directed graphs. *CoRR*, abs/1812.01768, 2018. arXiv:1812.01768.

11 Michel X. Goemans. Minimum bounded degree spanning trees. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '06, pages 273–282, Washington, DC, USA, 2006. IEEE Computer Society. doi:10.1109/FOCS.2006.48.

12 Fabrizio Grandoni and Bundit Laekhanukit. Surviving in directed graphs: a quasi-polynomial-time polylogarithmic approximation for two-connected directed steiner tree. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 420–428, 2017. doi:10.1145/3055399.3055445.

13 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. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019*, pages 253–264, 2019. doi:10.1145/3313276.3316349.

14 Mohammad Taghi Hajiaghayi. Open problems on bounded-degree network design from 8-th workshop on flexible network design, amsterdam, 2016. Announcement, 2016.

15 Eran Halperin and Robert Krauthgamer. Polylogarithmic inapproximability. In Lawrence L. Larmore and Michel X. Goemans, editors, *Proceedings of the 35th Annual ACM Symposium on Theory of Computing, June 9-11, 2003, San Diego, CA, USA*, pages 585–594. ACM, 2003. doi:10.1145/780542.780628.

16 Jochen Könemann and R. Ravi. A matter of degree: Improved approximation algorithms for degree-bounded minimum spanning trees. *SIAM J. Comput.*, 31(6):1783–1793, 2002. doi:10.1137/S009753970036917X.

17 Jochen Könemann and R. Ravi. Quasi-polynomial time approximation algorithm for low-degree minimum-cost steiner trees. In *FST TCS 2003: Foundations of Software Technology and Theoretical Computer Science, 23rd Conference, Mumbai, India, December 15-17, 2003, Proceedings*, pages 289–301, 2003. doi:10.1007/978-3-540-24597-1_25.

18 Jochen Könemann and R. Ravi. Primal-dual meets local search: Approximating msts with nonuniform degree bounds. *SIAM J. Comput.*, 34(3):763–773, 2005. doi:10.1137/S0097539702418048.

19 Guy Kortsarz and Zeev Nutov. Bounded degree group steiner tree problems. In *IWOCA'20, to appear*, 2020.

20 Lap Chi Lau, Joseph Naor, Mohammad R. Salavatipour, and Mohit Singh. Survivable network design with degree or order constraints. *SIAM J. Comput.*, 39(3):1062–1087, 2009. doi:10.1137/070700620.

21 Lap Chi Lau and Mohit Singh. Additive approximation for bounded degree survivable network design. *SIAM J. Comput.*, 42(6):2217–2242, 2013. doi:10.1137/110854461.

22 R. Ravi, Madhav V. Marathe, S. S. Ravi, Daniel J. Rosenkrantz, and Harry B. Hunt III. Many birds with one stone: multi-objective approximation algorithms. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 438–447, 1993. doi:10.1145/167088.167209.

23 Thomas Rothvoß. Directed steiner tree and the lasserre hierarchy. *CoRR*, abs/1111.5473, 2011. arXiv:1111.5473.

24 Mohit Singh and Lap Chi Lau. Approximating minimum bounded degree spanning trees to within one of optimal. *J. ACM*, 62(1):1:1–1:19, 2015. doi:10.1145/2629366.

25 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997. doi:10.1007/BF02523690.

A Property (6c) of Theorem 6: Concentration Bound on Number of Copies of a Vertex Appearing in T

Finally, we prove Property (6c) in Theorem 6. To this end, we shall fix a vertex $v \in V$. For every vertex $p \in \mathbf{V}^\circ$, let $z_p = \sum_{o \in \Lambda^*_{\mathbf{T}^\circ}(p) \cap \mathbf{O}_v} x_o$. By Constraint (5), we have $z_p \leq x_p$. Let $m_p = |\Lambda^*_{\mathbf{T}^\circ}(p) \cap \mathbf{O}_v \cap \mathbf{V}|$ be the total number of nodes in $\Lambda^*_{\mathbf{T}^\circ}(p) \cap \mathbf{O}_v$ that are selected by the rounding algorithm.

As is typical, we shall introduce a parameter $s > 0$ and consider the expectation the random exponential variables $\mathbf{e}^{sm_p}$ (we use $\mathbf{e}$ for the natural constant). We shall bound $\mathbb{E}[\mathbf{e}^{sm_p} | p \in \mathbf{V}]$ from bottom to top by induction. So, in this proof, it is more convenient to for us to use a different definition of levels: the level of a node p in $\mathbf{T}^\circ$ is the maximum number of edges in a path in $\mathbf{T}^\circ$ starting from p. So, the leaves have level 0 and for an internal node p in $\mathbf{T}^\circ$, the level of p is 1 plus the maximum of the level of q over all children q of p. We define an α_i for every integer $i \geq 0$ as $\alpha_0 = \mathbf{e}^s$ and $\alpha^i = \mathbf{e}^{\alpha_{i-1}-1}, \forall i \geq 1$. Notice that $\alpha_0, \alpha_1, \ldots$ is an increasing sequence. Thus, we can induce the following lemma.

▶ **Lemma 24.** *For any node p in $\mathbf{T}^\circ$ of level at most i, $\mathbb{E}\left[\mathbf{e}^{sm_p} \middle| p \in \mathbf{V}\right] \leq \alpha_i^{z_p/x_p}$.*

Proof. We prove the lemma by induction on i. If $i = 0$, then p is a leaf, and thus, we have either $z_p = 0$ or $z_p = x_p$, depending on whether $p \in \mathbf{O}_v$ or not. If $z_p = 0$, then m_p is always 0, and thus, $\mathbb{E}\left[\mathbf{e}^{sm_p} | p \in \mathbf{V}\right] = 1 = \alpha_0^{z_p/x_p}$. If $z_p = x_p$, then m_p is always 1 (conditioned on $p \in \mathbf{V}$), and thus, $\mathbb{E}\left[\mathbf{e}^{sm_p} | p \in \mathbf{V}\right] = \mathbf{e}^s = \alpha_0^{z_p/x_p}$. So, the lemma holds if $i = 0$.

Now, let $i \geq 1$ be any integer and we assume the lemma holds for $i-1$. We shall prove that it also holds for i. Focus on a node p of level at most i. Then all children q of p have level at most $i-1$. If p is a virtual node, then $p \in \mathbf{V}$ implies that both children of p in $\mathbf{V}$. Since the two children are handled independently in the rounding algorithm, we have

$$\begin{aligned}\mathbb{E}\left[\mathbf{e}^{sm_p} | p \in \mathbf{V}\right] &= \prod_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \mathbb{E}\left[\mathbf{e}^{sm_q} | p \in \mathbf{V}\right] = \prod_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \left[\frac{x_q}{x_p} \cdot \mathbb{E}[\mathbf{e}^{sm_q} | q \in \mathbf{V}] + 1 - \frac{x_q}{x_p}\right] \\ &= \prod_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \left[1 + \frac{x_q}{x_p}\Big(\mathbb{E}[\mathbf{e}^{sm_q} | q \in \mathbf{V}] - 1\Big)\right].\end{aligned}$$

If p is the super node or a state node, then we have $\sum_{q \in \Lambda_{\mathbf{T}^\circ}(p)} x_q = x_p$. Conditioned on $p \in \mathbf{V}$, the rounding procedure adds exactly one child q of p to $\mathbf{V}$. Then, we have

$$\begin{aligned}\mathbb{E}\left[\mathbf{e}^{sm_p} | p \in \mathbf{V}\right] &= \sum_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \frac{x_q}{x_p} \mathbb{E}\left[\mathbf{e}^{sm_q} | q \in \mathbf{V}\right] = 1 + \sum_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \frac{x_q}{x_p}\Big(\mathbb{E}[\mathbf{e}^{sm_q} | q \in \mathbf{V}] - 1\Big) \\ &\leq \prod_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \left[1 + \frac{x_q}{x_p}\Big(\mathbb{E}[\mathbf{e}^{sm_q} | q \in \mathbf{V}] - 1\Big)\right].\end{aligned}$$

Thus, we always have

$$\begin{aligned}&\mathbb{E}\left[\mathbf{e}^{sm_p} | p \in \mathbf{V}\right] \\ &\leq \prod_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \left[1 + \frac{x_q}{x_p}\Big(\mathbb{E}[\mathbf{e}^{sm_q} | q \in \mathbf{V}] - 1\Big)\right] \\ &\leq \prod_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \left[1 + \frac{x_q}{x_p}\big(\alpha_{i-1}^{z_q/x_q} - 1\big)\right] && \text{by induction hypothesis} \\ &\leq \exp\left[\sum_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \frac{x_q}{x_p}\big(\alpha_{i-1}^{z_q/x_q} - 1\big)\right] \\ &\leq \exp\left[\frac{z_p}{x_p}(\alpha_{i-1} - 1)\right] = \alpha_i^{z_p/x_p}. && \text{since } 1 + \theta \leq e^\theta \text{ for every } \theta\end{aligned}$$

To see the second inequality in the last line, we notice the following three facts: (i) $\alpha_{i-1}^\theta - 1$ is a convex function of θ and when $\theta = 0$ its value is 0, (ii) $z_q/x_q \in [0,1]$ for every q in the summation, and (iii) $\sum_{q \in \Lambda_{\mathbf{T}^\circ}(p)} \frac{x_q}{x_p} \cdot \frac{z_q}{x_q} = \frac{z_p}{x_p}$. So, the quantity inside $\exp(\cdot)$ has maximum value $\frac{z_p}{x_p}(\alpha_{i-1}^1 - 1)$. The equality in the last line is by the definition of α_i. ◀

Let $h' = \Theta(h) = \Theta(\log n)$ be the level of the root. Now, we set $s = \ln(1 + \frac{1}{2h'})$. We prove inductively the following lemma:

▶ **Lemma 25.** *For every $i \in [0, h']$, we have $\alpha_i \leq 1 + \frac{1}{2h'-i}$.*

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

$$\begin{aligned}\alpha_i = \mathbf{e}^{\alpha_{i-1}-1} &\le \mathbf{e}^{1+\frac{1}{2h'-i+1}} \le 1 + \frac{1}{2h'-i+1} + \left(\frac{1}{2h'-i+1}\right)^2 \\ &= 1 + \frac{2h'-i+2}{(2h'-i+1)^2} \le 1 + \frac{1}{2h'-i}.\end{aligned}$$

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

So, by Lemma 24 and 25, we have $\mathbb{E}[\mathbf{e}^{sm_{\mathbf{r}}}] \le \alpha^1_{h'} \le 1 + \frac{1}{h'} = 1 + O\left(\frac{1}{\log n}\right)$. This finishes the proof of Property (6c) in Theorem 6.

B Omitted Proofs

▶ **Lemma 3.** *Let $T = (V_T, E_T)$ be an n-vertex binary tree. Then there exists a vertex $v \in V_T$ with $n/3 < |\Lambda^*_T(v)| \le 2n/3 + 1$.*

Proof. We assume $n \ge 4$; otherwise, if $n = 3$, then we have $2n/3 + 1 = 3$, and $\mathrm{root}(T)$ satisfies the condition. Our goal is to find a vertex u with $n/3 < |\Lambda^*(u)| \le 2n/3 + 1$. Start from $u = \mathrm{root}(T)$ in the tree, and thus, we have $\Lambda^*(u) > 2n/3 + 1$. Let v be the child of u with the biggest $|\Lambda^*(v)|$. So, $|\Lambda^*(v)| \ge (|\Lambda^*(u)| - 1)/2 > n/3$. We then replace u with v. So $|\lambda^*(u)|$ has decreased but the condition $|\Lambda^*(u)| > n/3$ is maintained. Thus, if we repeat the process, we will eventually find a u with $n/3 < |\Lambda^*(u)| \le 2n/3 + 1$. ◀

▶ **Lemma 22.** *For a vertex u with $\ell_u \ge \gamma$, we have* $\Pr\left[\bigvee_{o \in O_t \cap \Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right] \ge \frac{1}{2(L+1-\ell_u)} \frac{z_u}{x_u}$.

Proof. There are two different approaches to prove the lemma, one based on bounding the conditional second moment of the random variable for the number of chosen vertices in $O_t \cap \Lambda^*_u$, and the other based on the mathematical induction on ℓ_u, which is the one we use here.

Suppose u is a leaf. Then $z_u/x_u = 1$ if $u \in O_t$ and $z_u/x_u = 0$ otherwise. So, we have $\Pr\left[\bigvee_{o \in O_t \cap \Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right] = \frac{z_u}{x_u}$ and the lemma clearly holds since we have $\ell_u \le L$.

Then, we prove the lemma by induction on ℓ_u. If $\ell_u = L$ then u must be a leaf and thus the lemma holds. We assume the lemma holds for every u with $\ell_u = \ell + 1$, for some $\ell \in [\gamma, L-1]$. Then we prove the lemma for u with $\ell_u = \ell$. If u is a leaf the lemma holds and thus we assume u is not a leaf.

$$\begin{aligned}&\Pr\left[\bigvee_{o \in O_t \cap \Lambda^*_u} \mathbf{E}_o \middle| \mathbf{E}_u\right] \\ &\ge 1 - \prod_{v \in \Lambda_u} \left(1 - \frac{x'_v}{x'_u} \cdot \frac{1}{2(L-\ell)} \frac{z_v}{x_v}\right) = 1 - \prod_{v \in \Lambda_u} \left(1 - \frac{x_v}{x_u} \cdot \frac{1}{2(L-\ell)} \cdot \frac{z_v}{x_v}\right) \\ &\ge 1 - \prod_{v \in \Lambda_u} \exp\left(-\frac{1}{2(L-\ell)} \cdot \frac{z_v}{x_u}\right) = 1 - \exp\left(-\frac{1}{2(L-\ell)} \cdot \frac{z_u}{x_u}\right) \\ &\ge \frac{1}{2(L-\ell)} \cdot \frac{z_u}{x_u} - \frac{1}{2}\left(\frac{1}{2(L-\ell)} \cdot \frac{z_u}{x_u}\right)^2 \ge \frac{1}{2(L-\ell)} \cdot \frac{z_u}{x_u} - \left(\frac{1}{2(L-\ell)}\right)^2 \frac{z_u}{x_u} \\ &= \left(\frac{2(L-\ell)-1}{(2(L-\ell))^2}\right) \frac{z_u}{x_u} \ge \frac{1}{2(L+1-\ell)} \cdot \frac{z_u}{x_u}.\end{aligned}$$

The inequality in the first line used the induction hypothesis: $\frac{x'_v}{x'_u}$ is the probability that we choose v in T_1 conditioned on that we choose u, and $\frac{1}{2(L-\ell)}\frac{z_v}{x_v}$ is the lower bound on the probability that we choose some vertex in $O_t \cap \Lambda_v^*$ conditioned on that v is chosen. The equality in the line used that $x'_u = 2^\gamma x_u$ and $x'_v = 2^\gamma x_v$. The inequality in the second line used that $1-\theta \le e^{-\theta}$ for every real number θ. The first inequality in the third line used that $e^{-\theta} \le 1-\theta+\frac{\theta^2}{2}$ for every $\theta \ge 0$. The second inequality in the line used Property 4, which says $\frac{z_u}{x_u} \le 2$. The last inequality used that $(2(L-\ell)-1)\cdot 2(L-\ell+1) \ge 4(L-\ell)^2$ since $L-\ell \ge 1$. ◀

▶ **Lemma 23.** *For every vertex $\ell_u = \ell \le \gamma$, we have* $\Pr\left[\bigvee_{o\in O_t\cap\Lambda_u^*} \mathbf{E}_o \middle| \mathbf{E}_u\right] \ge \alpha_\ell \frac{z_u}{x_u}$.

Proof. We prove this lemma via mathematical induction. The lemma holds if $\ell = \gamma$ as we mentioned. So, we assume $\ell < \gamma$ and the lemma holds with ℓ replaced by $\ell+1$. If u is a leaf, then we have $\Pr\left[\bigvee_{o\in O_t\cap\Lambda_u^*} \mathbf{E}_o \middle| \mathbf{E}_u\right] = \frac{z_u}{x_u}$ and the lemma holds. So, again we assume u is not a leaf. Then,

$$\begin{aligned}
\Pr\left[\bigvee_{o\in O_t\cap\Lambda_u^*} \mathbf{E}_o \middle| \mathbf{E}_u\right] &\ge 1 - \prod_{v\in\Lambda_u}\left(1-\frac{x'_v}{x'_u}\alpha_{\ell+1}\frac{z_v}{x_v}\right) = 1-\prod_{v\in\Lambda_u}\left(1-\frac{2x_v}{x_u}\alpha_{\ell+1}\frac{z_v}{x_v}\right)\\
&\ge 1-\prod_{v\in\Lambda_u}\exp\left(-2\alpha_{\ell+1}\frac{z_v}{x_u}\right) = 1-\exp\left(-2\alpha_{\ell+1}\frac{z_u}{x_u}\right)\\
&\ge 2\alpha_{\ell+1}\frac{z_u}{x_u} - \frac{1}{2}\left(2\alpha_{\ell+1}\frac{z_u}{x_u}\right)^2 \ge 2\alpha_{\ell+1}\frac{z_u}{x_u} - (2\alpha_{\ell+1})^2\frac{z_u}{x_u} = \alpha_\ell\frac{z_u}{x_u}.
\end{aligned}$$

To see the equality in the first line, we notice that $x'_u = 2^\ell x_u$ and $x'_v = 2^{\ell+1}x_v$ for every $v \in \Lambda_u$. Many other inequalities used the same arguments as in Lemma 22. ◀

Permutation Strikes Back: The Power of Recourse in Online Metric Matching

Varun Gupta
University of Chicago, IL, USA
guptav@uchicago.edu

Ravishankar Krishnaswamy
Microsoft Research India, Bangalore, India
rakri@microsoft.com

Sai Sandeep
Carnegie Mellon University, Pittsburgh, PA, USA
spallerl@andrew.cmu.edu

Abstract

In this paper, we study the online metric matching with recourse (OMM-Recourse) problem. Given a metric space with k servers, a sequence of clients is revealed online. A client must be matched to an available server on arrival. Unlike the classical online matching model where the match is irrevocable, the recourse model permits the algorithm to rematch existing clients upon the arrival of a new client. The goal is to maintain an online matching with a near-optimal total cost, while at the same time not rematching too many clients.

For the classical online metric matching problem without recourse, the optimal competitive ratio for deterministic algorithms is $2k-1$, and the best-known randomized algorithms have competitive ratio $O(\log^2 k)$. For the much-studied special case of line metric, the best-known algorithms have competitive ratios of $O(\log k)$. Improving these competitive ratios (or showing lower bounds) are important open problems in this line of work.

In this paper, we show that logarithmic recourse significantly improves the quality of matchings we can maintain online. For general metrics, we show a deterministic $O(\log k)$-competitive algorithm, with $O(\log k)$ recourse per client, an exponential improvement over the $2k-1$ lower bound without recourse. For line metrics we show a deterministic 3-competitive algorithm with $O(\log k)$ amortized recourse, again improving the best-known $O(\log k)$-competitive algorithms without recourse. The first result (general metrics) simulates a batched version of the classical algorithm for OMM called Permutation. The second result (line metric) also uses Permutation as the foundation but makes non-trivial changes to the matching to balance the competitive ratio and recourse.

Finally, we also consider the model when both clients and servers may arrive or depart dynamically, and exhibit a simple randomized $O(\log n)$-competitive algorithm with $O(\log \Delta)$ recourse, where n and Δ are the number of points and the aspect ratio of the underlying metric. We remark that no non-trivial bounds are possible in this fully-dynamic model when no recourse is allowed.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases online algorithms, bipartite matching

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.40

Category APPROX

Related Version A full version of the paper is available at `https://arxiv.org/abs/1911.12778`.

1 Introduction

The classical online metric matching (OMM) problem is defined on a metric space $(\mathcal{X}, d)$, where $\mathcal{X}$ denotes a set of n points where the servers and clients are located, and a distance function (metric) $d : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_+$. A set $S \subseteq \mathcal{X}$ of servers, $|S| = k$, is given offline, and a


© Varun Gupta, Ravishankar Krishnaswamy, and Sai Sandeep;
licensed under Creative Commons License CC-BY

sequence of client requests $C = (c_1, \ldots, c_k)$ is revealed in an online manner. The algorithm is required to match each client request to an available (previously unmatched) server on arrival irrevocably. The objective is to minimize the total cost of the matching, which is the sum of distances between matched client-server pairs. The performance of an algorithm is measured using the notion of competitive ratio – the worst-case over all instances of the ratio of the cost of the online algorithm and the cost of an optimal offline matching.

This problem was first studied in two independent works [17, 18]. Both these works present a $(2k-1)$-competitive deterministic algorithm called PERMUTATION, and also show that *this bound is tight among deterministic algorithms.* Subsequently, the work of [22] shows that randomization can overcome this lower bound (for oblivious adversaries) by giving a $\mathcal{O}(\log^3 k)$-competitive[1] randomized algorithm, which was later improved in [4], where the authors show an $\mathcal{O}(\log^2 k)$-competitive algorithm. In contrast, the best known lower bound for randomized algorithms is a factor of $\Omega(\log k)$[22]. Resolving this gap has remained a challenging and long-standing open problem in this area.

The OMM problem has also elicited much interest in specialized metrics such as the line metric (OMM-LINE). It was long conjectured that OMM-LINE should admit a 9-competitive algorithm, by virtue of the connections between this problem and another classical problem in online algorithms known as the cow-path problem. However, [10] disproved this particular conjecture by presenting a lower bound of 9.001. In terms of upper bounds, no algorithms with better competitive ratios than those for general metrics were known until a recent line of work [15, 23, 25] gave improved algorithms for the line metric. The current best algorithm is a deterministic $\mathcal{O}(\log k)$-competitive algorithm [25]. It is still an open question whether constant-competitive algorithms exist for OMM-LINE. Intriguingly, there are $\Omega(\log k)$ lower bounds on natural families of algorithms [1, 19].

Given these barriers for OMM, we study whether we can obtain strictly better performance if we are *allowed to re-match a few clients* upon the arrival of a new client.

▶ **Problem 1** (OMM-RECOURSE)**.** *An instance consists of a metric space $(\mathcal{X}, d)$, and a set $S \subseteq X$ of servers with $|S| = k$. A sequence of client requests $C_k = (c_1, \ldots, c_k)$ is revealed in an online manner. At time t, after the algorithm observes c_t, it must maintain a matching $\mathcal{M}_t$ such that every client is matched to exactly one server, and each server is matched to at most one client. The algorithm can re-match some earlier clients, and the number of times clients are re-matched is called the recourse.*

▶ **Definition 2.** *We say that an online algorithm is α-competitive with β-amortized recourse for* OMM-RECOURSE *if for all $t \in [k]$, the cost of the algorithm's matching for $C_t := (c_1, \ldots, c_t)$ is at most α times the cost of the optimal matching for C_t, and the total number of recourse steps taken so far is at most βt. Additionally, the algorithm is said to have a per-client recourse of β if no client is rematched more than β times.*

While our main theoretical motivation is in understanding the power of recourse in the classical OMM problem, often it is also the case in practice that matching decisions *are not irrevocable*, and instead, there is simply a cost (or) penalty for re-assignments. For example, in a video streaming setting, users arrive online and want to stream a video. The ISP must choose a server to stream from, preferring a server closer to the user. Of course, this decision can be changed over the time horizon, but this will cause a temporary disruption that must be minimized. The recourse model then naturally captures the competing goals of minimizing cost and the number of re-assignments. Moreover, the stronger notion of *per-client* recourse guarantees a fairness property by bounding the inconvenience for each client.

[1] Throughout the paper, logarithms are with respect to base 2.

The main results of this paper affirmatively answer the question of whether limited recourse can help in the OMM problem.

▶ **Theorem 3.** *There is an efficient deterministic* $2 \log k$*-competitive algorithm with* $\log k$ *per-client recourse for* OMM-Recourse *on general metrics. (Section 3)*

Theorem 3 coupled with the $(2k - 1)$ lower bound for deterministic algorithms without recourse highlights the exponential improvement in competitive ratio possible with limited recourse. We complement the above result by noting that the guarantees given above are tight for our algorithm. We also show that *any deterministic algorithm* which has constant per-client recourse must have a logarithmic competitive ratio.

▶ **Theorem 4.** *No deterministic algorithm for* OMM-Recourse *with per-client recourse at most an absolute constant* C *can have a competitive ratio* $o(\log k)$*. (Appendix C)*

Our algorithm for Theorem 3 is an adaptation of the classical Permutation algorithm for OMM [17, 18] which recomputes the offline optimal matching on the arrival of a new client, and matches the arriving client to the server that is part of the new offline matching but as yet unmatched in the online matching. While Permutation is a very natural and elegant algorithm, it fares poorly in the standard model without recourse where it has a competitive ratio of $\Omega(k)$ *even on line metrics*! At a high level, the worst-case behavior for the classical Permutation algorithm for OMM occurs when clients arrive one-by-one, and the competitive ratio improves if the clients arrive in batches. Our algorithm then *uses recourse to mimic* the output of such a batched version of Permutation.

We next turn our attention to the special case of the line metric, where we present a special-purpose algorithm that significantly improves on Theorem 3:

▶ **Theorem 5.** *There is a deterministic* 3*-competitive algorithm with* $\mathcal{O}(\log k)$*-amortized recourse for* OMM-Line-Recourse*. (Section 4)*

Note that obtaining constant-competitive algorithms for the line metric has been a challenging open problem for OMM– our algorithm achieves such a guarantee when equipped with a small amount of recourse. This result is also the main technical contribution of our work. Once again, our algorithm builds on the classical Permutation algorithm for OMM. Exploiting the nature of the line metric, we view the online matching determined by Permutation as a collection of directed forward or backward arcs from the clients to servers. Noting that such a matching may be sub-optimal only if *overlapping* forward and backward arcs are present, our algorithm tries to re-match some clients to eliminate such overlaps. However, blindly re-matching to eliminate all overlapping arcs can lead to a large recourse. Instead, we propose a clever *asymmetric uncrossing method* to balance the competitive ratio and recourse. Our analysis is somewhat non-standard in that we first identify a *family of algorithms* for OMM-Line-Recourse, all of which share the same asymmetric uncrossing criterion, and incur the same cost. Of this family, we first choose one algorithm (whose cost analysis is most intuitive) to bound the competitive ratio of *all the algorithms* in the family. However, this algorithm can demonstrably incur large recourse. Hence, our actual algorithm is another one from this family which is designed for minimizing the recourse but whose cost analysis in a direct manner is not apparent to us.

Finally, we focus on another limitation of classical OMM– due to the irrevocable nature of assignments, the competitive ratio would be unbounded when both clients and/or servers can arrive or depart the system. Hence, the classical model only considers the setting when all servers are known ahead of time and clients arrive in an online manner. We show that by allowing recourse, we can, in fact, handle arrivals and departures of clients and servers:

▶ **Theorem 6.** *There is a randomized $\mathcal{O}(\log n)$-competitive algorithm with $\mathcal{O}(\log \Delta)$ amortized recourse for* OMM-Recourse *when clients and servers can arrive and depart, where Δ is the aspect ratio of the metric space.*

We defer the details of this dynamic model to [16].

Related Work. To the best of our knowledge, the only work which considers recourse for online min-cost matching is the recent work [20] where the authors consider a *two-stage* version of the uni-chromatic problem (where there is no distinction between servers and clients): In the first stage, a perfect matching between $2n$ given nodes must be selected; in the second stage $2k$ new nodes are introduced. The goal is to produce α-competitive matchings at the end of both stages, and such that the number of edges removed from the first stage matching is at most βk. The authors show that $\alpha = 3, \beta = 1$ and $\alpha = 10, \beta = 2$ are possible when k is known or unknown, respectively. Our results can be seen as a multi-stage generalization of this two-stage model, although the two models are slightly different in terms of the distinction between servers and clients.

A related model capturing a different kind of flexibility in online matching is that of *matching with delays* [8, 5]: here, the requests need not be matched at the time of arrival, but accrue a delay penalty until the algorithm matches them. The algorithm must minimize the total matching cost plus total delay penalty. The current best known randomized algorithms are $\mathcal{O}(\log n)$ competitive [2], which also proves a lower bound of $\Omega\left(\frac{\log n}{\log \log n}\right)$. The best known deterministic algorithms are $\mathcal{O}(k^{0.59})$-competitive [3]. Another class of beyond-worst case models are stochastic models, such as *i.i.d.* and *random order* settings. The majority of work in this vein has been done in the reward maximization objective (see e.g., [11, 7, 6] and references therein). For OMM, [24] gave a deterministic algorithm that is simultaneously $\mathcal{O}(\log k)$-competitive in the random order model and $(2k-1)$-competitive in worst case. Recently, [12] show $\mathcal{O}((\log \log \log k)^2)$-competitive algorithms in the known *i.i.d.* model.

Finally, online algorithms with recourse have also been studied in other settings such as scheduling and set cover (see, e.g., [14, 13, 9] and the references within.).

Recently, and independent of our work, [21] have obtained an $\mathcal{O}(1)$-competitive algorithm with $\mathcal{O}(\log k)$-amortized recourse for OMM-Line-Recourse with a very different approach of extending the t-net framework of [25].

2 Preliminaries

For most of the paper (except for the fully dynamic setting), we consider the setting where the servers $\mathcal{S}$ are known up front. The clients arrive online, and we denote by $C_t = (c_1, \ldots, c_t)$ the set of the first t clients. An optimal matching between C_t and $\mathcal{S}$ is denoted by $\mathcal{M}_t^*$, and similarly, the algorithm's matching between C_t and $\mathcal{S}$ will be denoted by $\mathcal{M}_t$. We denote by OPT_t, the cost of the optimal matching $\mathcal{M}_t^*$. For any matching $\mathcal{M}$, we use $\mathcal{M}(c)$ and $\mathcal{M}(C)$ to denote the server and the set of servers matched to the client c and the set of clients C, respectively. We define $\mathcal{M}(s)$ and $\mathcal{M}(S)$ similarly.

The Permutation algorithm. As mentioned in Section 1, [17] and [18] independently proposed a $(2k-1)$-competitive algorithm Permutation for OMM. Since our algorithms build extensively on this algorithm, we first summarize Permutation and its key properties. The algorithm maintains two matchings: the current online matching $\mathcal{M}_t$, and the optimal offline matching $\mathcal{M}_t^*$ of the clients C_t that have arrived so far. The main observation behind

Algorithm 1 PERMUTATION (metric $(\mathcal{X}, d)$, server-optimal matching $\mathcal{M}_{t-1}$ for C_{t-1}).

for new batch of clients $C_{\text{cur}} = C_{t+\ell} \setminus C_{t-1} = \{c_t, c_{t+1}, \ldots, c_{t+\ell}\}$ that arrives **do**
  let $\mathcal{M}^*_{t-1}$ and $\mathcal{M}^*_{t+\ell}$ be optimal matchings for C_{t-1} and $C_{t+\ell}$ from Lemma 7.
  let $S_{\text{cur}} = S^*_{t+\ell} \setminus S^*_{t-1}$ be the set of $\ell+1$ servers matched in $\mathcal{M}^*_{t+\ell}$ but not in $\mathcal{M}^*_{t-1}$
  let $\mathcal{M}_{\text{cur}}$ denote the minimum cost matching between C_{cur} and S_{cur}
  augment $\mathcal{M}_{t-1}$ using $\mathcal{M}_{\text{cur}}$ to obtain the new matching $\mathcal{M}_{t+\ell}$
end for

the algorithm is that, when a new client c_{t+1} arrives, there exists an optimal matching of C_{t+1} to $\mathcal{S}$ which uses *exactly the servers used in $\mathcal{M}^*_t$ plus one extra server*. PERMUTATION simply identifies the extra server s_{t+1} and matches c_{t+1} with s_{t+1}. This property can be formalized as follows.

▶ **Lemma 7** (Lemma 2.2 in [17]). *There exists a sequence of optimal matchings $\mathcal{M}^*_1, \ldots, \mathcal{M}^*_k$ matching client sets $C_1, \ldots, C_k$ to $\mathcal{S}$ such that the sets of servers used in these matchings, $S^*_i := \mathcal{M}^*_i(C_i)$ are nested, i.e., $S^*_1 \subseteq S^*_2 \subseteq \cdots \subseteq S^*_k$.*

▶ **Definition 8** (Server-optimal matching). *At time t, a matching $\mathcal{M}_t$ of client-set C_t is said to be* server-optimal *if it uses the same servers as $\mathcal{M}^*_t$, i.e., $\mathcal{M}^*_t(C_t) = \mathcal{M}_t(C_t)$.*

▶ **Proposition 9** ([17]). PERMUTATION *always maintains a server-optimal matching.*

The notion of server-optimality will be crucial throughout this paper. Intuitively, it says that the algorithm has identified the right set of servers from $\mathcal{S}$ to match to, and is only sub-optimal w.r.t to the actual matching maintained between C_t and $\mathcal{M}_t(C_t)$.

Algorithm 1 gives a more general version of PERMUTATION which we will use later, where clients arrive in *batches*, and we add a *minimum-cost matching* between the arriving clients in the batch and the additional servers an optimal solution uses (from Lemma 7).

▶ **Lemma 10** ([17]). *After the arrival of a batch of clients $C_{t+\ell} \setminus C_{t-1}$, the cost of the matching $\mathcal{M}_{\text{cur}}$ computed in Algorithm 1 is at most $2\mathit{OPT}_{t+\ell}$.*

▶ **Theorem 11** (Theorem 2.4 in [17]). *Algorithm 1 is $(2m-1)$-competitive for online weighted matching if the requests arrive in m batches.*

3 Online matching with recourse for general metrics

In this section, we prove Theorem 3 by showing that Algorithm 2 is a $(2\log k, \log k)$-competitive algorithm for OMM-RECOURSE in a general metric. To motivate the algorithm, note that Theorem 11 says that in order to minimize competitive ratio, it is best to feed the client sequence to PERMUTATION in as few batches as possible. However, we are also constrained in matching clients immediately on arrival. One way of balancing the two goals is to run PERMUTATION incrementally on each client arrival, and use recourse to periodically unmatch a suffix of client sequence and re-introduce these clients as *a single batch*. As an example, assume that we create B batches of k/B clients, with the jth batch consisting of clients $\text{Batch}_j = \{(j-1)k/B+1, \ldots, jk/B\}$. As clients in batch j arrive, we first match them via vanilla PERMUTATION. After the (jk/B)th client arrives, we unmatch all clients in Batch_j and re-introduce them as one single batch. The amortized recourse of this algorithm is 1. Moreover, the matching at any time t may be viewed as the output of running PERMUTATION with at most B batches of k/B clients each and k/B batches of 1 client each. Setting $B = \sqrt{k}$ then gives us an $O(\sqrt{k})$-competitive algorithm with a per-client recourse of 1!

Algorithm 2 MultiScalePermutation (metric (X, d) and server set $\mathcal{S} \subseteq X$).

initialize matching $\mathcal{M}_0 = \emptyset$
for each new client c_t that arrives at time-step t **do**
  let $i(t) = \arg\max_i$ s.t t is divisible by 2^i
  unmatch the latest $2^{i(t)}$ clients $\{c_{t-2^{i(t)}+1}, \ldots, c_t\}$ and revert to matching $\mathcal{M}_{t-2^{i(t)}}$
  introduce a block of clients $\{c_{t-2^{i(t)}+1}, \ldots, c_t\}$ to Algorithm 1 with the current matching being $\mathcal{M}_{t-2^{i(t)}}$ and update $\mathcal{M}_t$ to be the resulting matching for the clients C_t
end for

To get a smaller competitive ratio at the expense of slightly higher recourse, we employ the following natural extension: imagine the first t clients as the leftmost t leaves of a balanced binary tree of depth $\lceil \log_2 k \rceil$. If an arriving client is the last leaf of some subtree, we unmatch all clients in the largest such subtree and re-introduce them as a single batch. The competitive ratio would then be bounded by $2 \log k$ since the matching at time t is simply the combination of at most $\log k$ batches (based on the binary decomposition of t) and by using Theorem 11. Further, any client is rematched at most $\log k$ times, since the size of the batch it is part of doubles on every rematch.

Proposition 12 generalizes the result in Theorem 3 to give a trade-off between the cost and recourse, and Proposition 13 proves that our analysis of Algorithm 2 is tight. The proofs of these propositions appear in Appendix B.

▶ **Proposition 12.** *Algorithm 2 with the constant 2 replaced by d, gives an $(d-1)\log_d k$-competitive algorithm with $\log_d k$-per client recourse. In particular, for any $d = \mathcal{O}(1)$ we get $\mathcal{O}(\log k)$-competitive algorithm with $\mathcal{O}(\log k)$-per client recourse, and for $d = k^\alpha$ ($\alpha \leq 1$), we get an $\tilde{\mathcal{O}}(k^\alpha)$-competitive algorithm with $1 + 1/\alpha$-per client recourse.*

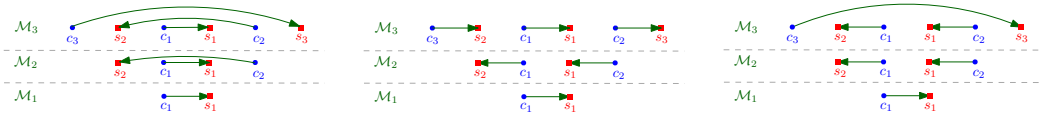
▶ **Proposition 13.** *The cost-recourse tradeoff of Theorem 3/Proposition 12 is tight: for $d = 2$, there is a sequence of instances where Algorithm 2 is $\Omega(\log k)$-competitive and has $\Omega(\log k)$ per-client recourse. Further, there is an increasing sequence $\{d_i\}$, such that with $d = d_i$ there is a sequence of instances where Algorithm 2 is $\Omega(d \log_d k)$-competitive with $\Omega(\log_d k)$ per-client recourse.*

4 Online Matching on the Line Metric

In this section, we focus on the special case of a line metric, where, for all points $x \in \mathcal{X}$, we associate a location $\ell : \mathcal{X} \to \mathbb{R}$ such that $d(x, y) = |\ell(x) - \ell(y)|$. We also assume without loss of generality that all the clients and servers are in distinct locations on the line.

Our starting point is again the Permutation algorithm which, by itself can have $\Omega(k)$ competitive ratio even on line metrics: see Figure 1a, where the distance between any consecutive client and server is 1[2]. Permutation would first match c_1 to s_1, and then c_2 to s_2 (the set $\{s_1, s_2\}$ is server-optimal as it admits an optimal matching $(c_1, s_2), (c_2, s_1)$). Continuing in this manner, Permutation would incur a total cost of $\Omega(k^2)$, whereas the optimal matching would have a cost of k. A natural fix would seem to be to re-match the clients and servers in the existing matching to maintain an optimal matching at all times, but as illustrated in the example in Figure 1b, this can lead to $\Omega(k)$ amortized recourse.

[2] Even though in this example it seems that Permutation can perform well by breaking ties correctly, we can modify the edge lengths very slightly to force the matchings, thus proving that no clever tie-breaking can help the algorithm.



(a) PERMUTATION without re-match.

(b) PERMUTATION with full re-match.

(c) *Asymmetric* PERMUTATION with re-match.

Figure 1 Illustrative examples of OMM-LINE-RECOURSE.

One of our main observations is that we can view a matching on a line metric as composed of *forward* and *backward* arcs between the matched clients and servers based on their relative location (as in Figure 1a), and, as we formalize later, a matching between a given client and server set is sub-optimal *if and only if* it contains overlapping forward and backward arcs. Indeed, PERMUTATION does no recourse but has large overlaps, and always re-matching overlapping arcs output by PERMUTATION yields an optimal solution but with large recourse. Our idea is to balance the overlap and recourse by *re-matching overlapping pairs asymmetrically*. When a new client c_t arrives, let s_t be the new server that PERMUTATION brings in to the system. Then, if (c_t, s_t) is a forward arc i.e. if $\ell(c_t) \leq \ell(s_t)$, our algorithm simply adds it as is, even if it overlaps with existing backward arcs. On the other hand, when (c_t, s_t) is a backward arc, we use rematches to *maximally cancel* portions of this backward arc with overlapping portions of existing forward arcs. See Figure 1c for an example where $\mathcal{M}_2$ has undergone a re-matching, while $\mathcal{M}_3$ has not.

While this re-matching process is unambiguous for the example in Figure 1c, in general, there could be multiple ways of re-matching overlapping arcs in $\mathcal{M}_{t-1} \cup \{c_t, s_t\}$. We, therefore, begin by defining a *family of asymmetric maximally canceling algorithms* in Section 4.1 and prove that all algorithms in this family incur the same cost. In Section 4.2, we study one special algorithm, RECURSIVECANCEL in this family which is the most amenable for cost analysis and bound the competitive ratio of the entire family of algorithms by 3. However, RECURSIVECANCEL can incur a large $\Omega(k)$ recourse, and so our final algorithm MINIMUMCANCEL in Section 4.3 further identifies a way of re-matching backward arcs which *minimally changes the existing matching*. Using this property, we show that the MINIMUMCANCEL algorithm has $O(\log k)$ amortized recourse, thereby proving Theorem 5.

Before going into the details of the algorithms, we first introduce a property of the PERMUTATION algorithm on the line metric that is useful:

▶ **Lemma 14.** *If* PERMUTATION *maintains edges* $(c_1, s_1), (c_2, s_2), \ldots, (c_t, s_t)$ *at time* t*, then any server* $s_{t'}$ *added by the algorithm at time* $t' > t$ *will lie outside all these arcs.*

The proof follows from the server optimality of the PERMUTATION algorithm and is presented in Appendix B.

4.1 Preliminary Concepts and Notation

We begin by introducing a few concepts that are important for cost analysis of matching on the line metric: forward and backward arcs, atomic intervals, and discrepancy. We end the section by defining the family of asymmetric maximally canceling algorithms. Unless otherwise stated, $C_t = \{c_1, c_2, \ldots, c_t\}$ will denote the set of clients which have arrived by time t, and $S_t = \{s_1, s_2, \ldots, s_t\}$ will denote the set of servers chosen by PERMUTATION.

▶ **Definition 15** (Forward and Backward Arcs). *Let a client* c *be matched to server* s *in a given matching* $\mathcal{M}$*. We call the edge* (c, s) *a* forward arc $\overrightarrow{c, s}$ *if* $\ell(c) \leq \ell(s)$ *and a* backward arc $\overleftarrow{s, c}$ *otherwise.*

We now note that in an optimal matching, no pair of forward and backward arcs cross.

▶ **Proposition 16.** *Consider a set C of clients and a set S of servers with $|C| = |S|$. Then, a matching $\mathcal{M}$ between C and S is optimal for the client-server set (C, S) if and only if, for any forward arc $\overrightarrow{c_1, s_1} \in \mathcal{M}$ and backward arc $\overleftarrow{s_2, c_2} \in \mathcal{M}$, the intervals $[\ell(c_1), \ell(s_1)]$ and $[\ell(s_2), \ell(c_2)]$ are disjoint.*

Now, to compute the cost of any matching $\mathcal{M}_t$ between $C_t \cup S_t$, our approach will be to decompose the total cost into the contribution of what we call *atomic intervals* corresponding to $C_t \cup S_t$. Informally, the atomic intervals partition the line into open intervals between *two consecutive points* in $C_t \cup S_t$ in the metric space.

▶ **Definition 17** (Atomic intervals)**.** *For two nodes $p_1, p_2 \in C_t \cup S_t$ in the matching $\mathcal{M}_t$ at time t, we call the open interval $I = (\ell(p_1), \ell(p_2))$ an* atomic interval *if and only if for any other $p \in C_t \cup S_t$, it holds that $\ell(p) \notin (\ell(p_1), \ell(p_2))$. We denote the set of all atomic intervals at time t by $\mathcal{AI}_t$. We denote by $|I| = |\ell(p_1) - \ell(p_2)|$ the length of the interval I.*

Note that $\mathcal{AI}_t$ depends only on the set $C_t \cup S_t$, and grows in t as this set expands. We call $I' = (\ell(p_1'), \ell(p_2'))$ a subinterval of $I = (\ell(p_1), \ell(p_2))$, denoted $I' \subseteq I$ if $\ell(p_1) \leq \ell(p_1') \leq \ell(p_2') \leq \ell(p_2)$.

▶ **Definition 18** (Discrepancy)**.** *For any atomic interval $I \in \mathcal{AI}_t$ with its left end-point at location l, we define its* discrepancy *at time t to be the excess number of servers to the left of l in the currently used set of servers S_t:* $\mathrm{disc}_t(I) := |S_t \cap (-\infty, l]| - |C_t \cap (-\infty, l]|$.

The next lemma shows that the discrepancy of atomic intervals immediately gives the cost of the optimal matching.

▶ **Lemma 19.** *At any time t, and for any atomic interval $I \in \mathcal{AI}_t$, exactly $|\mathrm{disc}_t(I)|$ arcs cross I in an optimal matching between C_t and S_t. Further, if $\mathrm{disc}_t(I)$ is positive (respectively, negative), the direction of the crossing arcs is backward (resp., forward). Consequently, the cost of an optimal matching is $\mathsf{OPT}_t = \sum_{I \in \mathcal{AI}_t} |\mathrm{disc}_t(I)| \cdot |I|$.*

While $\mathrm{disc}_t(I)$ quantifies the minimum number of arcs that must cross I in any feasible matching, there could be many more arcs crossing I in the sub-optimal matching maintained by our algorithm. To this end, for a given matching $\mathcal{M}_t$ between $C_t \cup S_t$, and any subinterval $I' \subseteq I \in \mathcal{AI}_t$, let $n_t^f(I')$ and $n_t^b(I')$ denote the number of forward and backward arcs, respectively, crossing I' at time t. The following is then easy to see.

▷ Claim 20. For a matching $\mathcal{M}_t$, and any atomic interval $I \in \mathcal{AI}_t$, we have $n_t^b(I) - n_t^f(I) = \mathrm{disc}_t(I)$. Furthermore, the cost of $\mathcal{M}_t$ is $\mathsf{cost}(\mathcal{M}_t) = \sum_{I \in \mathcal{AI}_t} |n_t^f(I) + n_t^b(I)| \cdot |I|$.

We are now ready to define the family of asymmetric maximally canceling algorithms for OMM-Line-Recourse.

▶ **Definition 21** (Asymmetric maximally canceling algorithm)**.** *We call an algorithm an* asymmetric maximally canceling algorithm *if the sequence of matchings $\{\mathcal{M}_t\}$ produced by the algorithm satisfies:*

1. *The server sets $\{S_t\}$ are given by* Permutation *(with a deterministic tie breaking rule);*
2. *Denoting s_t as the new server added by* Permutation *at time t:*
   - *If (c_t, s_t) is a forward arc, then no rematches are made. In other words, for all atomic intervals $I \in \mathcal{AI}_t$ which overlap with $(\ell(c_t), \ell(s_t))$ we have: $n_t^f(I) = n_{t-1}^f(I) + 1$.*

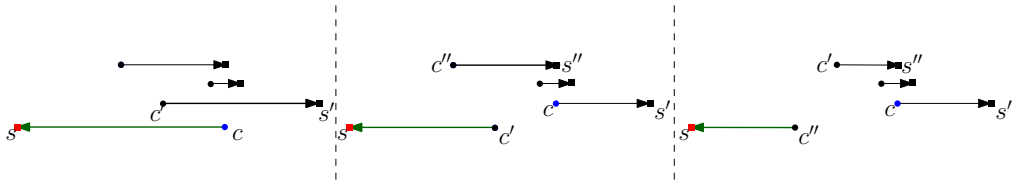


Figure 2 Illustration of RecursiveCancel: The leftmost figure shows the initial matching – (c, s) is the new backward arc and (c', s') is the forward arc chosen at step 9 of first iteration. The center figure shows the matching at step 11 after the rematching. The edge (c', s) becoems the new backward arc, and (c'', s'') is the forward edge picked in step 9 for the next iteration where (c', s) and (c'', s'') are replaced by (c'', s) and (c', s'') as in the rightmost figure.

- *If (c_t, s_t) is a backward arc, we use it to cancel existing forward arcs as much as possible. In other words, in the resulting matching $\mathcal{M}_t$ (after recourse), for all atomic intervals $I \in \mathcal{AI}_t$ which overlap with $(\ell(s_t), \ell(c_t))$ we have: $n^f_t(I) = \max\{n^f_{t-1}(I) - 1, 0\}$.*
- *$n^f_t(I)$ is unchanged for all atomic intervals that do not overlap with $(\ell(c_t), \ell(s_t))$.*

Note that it suffices to define just n^f_t as $n^b_t(I)$ can be derived from $n^f_t(I)$ for all I using Claim 20. Furthermore, the definition does not restrict how the rematches are performed, but only that all asymmetric maximally canceling algorithms have the same values of $n^f_t(I)$ and $n^b_t(I)$, for all the atomic intervals. Claim 20 immediately gives:

▷ Claim 22. All asymmetric maximally canceling algorithms incur the same matching cost.

The next section gives a bound on this cost by focusing on a specific member in this family by exploiting a useful invariant.

4.2 Algorithm RecursiveCancel for Bounding Cost

We now present our algorithm RecursiveCancel (Algorithm 3) and bound its cost. When a new backward arc overlapping with existing forward arcs is added, RecursiveCancel chooses the overlapping forward arc with the rightmost server and rematches the clients and servers of these two arcs to elimate overlap. This procedure results in at most one remnant backward arc (a strict suffix, i.e., server end, of the original backward arc) which can overlap with existing forward arcs. RecursiveCancel then recurses on this remnant until there is no overlap with a forward arc.

We begin with a couple of simple but useful observations about the behavior of rematches made by RecursiveCancel.

▶ **Lemma 23.** *Let (c, s) and (c', s') be two arcs defined in steps 8 and 9, respectively, of an iteration of* RecursiveCancel*. Then, (i) the new edge (c', s) added in step 11 is a backward arc, thereby proving that the while loop is well-defined; and (ii) the edge (c, s') added in step 10 is either a backward arc, or it is a suffix of the original forward arc (c', s').*

Proof. From Lemma 14, we know that there are no free servers inside an arc added by Permutation. Furthermore, since any arc in the current matching under RecursiveCancel is a sub-arc of some original arc added by Permutation, throughout the execution of RecursiveCancel algorithm there is never a free server inside an arc in the matching $\mathcal{M}_t$. Thus, c' is to the right of s, otherwise the server s which is free in $\mathcal{M}_{t-1}$, is inside the arc (c', s'). The second part of the lemma can be proved by noting that the forward arc (c', s') intersects the backward arc (c, s), and thus $\ell(c') < \ell(c)$. ◀

Algorithm 3 Algorithm RecursiveCancel.

1: set $\mathcal{M}_0 = \emptyset$
2: **for** each client c_t arriving at time $t \geq 1$ **do**
3: let s_t be the server Permutation matches c_t to, and let $a := (c_t, s_t)$
4: **if** a is a forward arc, i.e., $\ell(c_t) < \ell(s_t)$ **then**
5: $\mathcal{M}_t = \mathcal{M}_{t-1} \cup \{a\}$
6: **else** ▷ (c_t, s_t) is a backward arc
7: **while** there exists a forward arc in $\mathcal{M}_{t-1}$ which overlaps with a **do** ▷ a is the current backward arc
8: let $(c, s) := a$
9: let (c', s') be the forward arc overlapping with a with the *rightmost* server s'
10: $\mathcal{M}_{t-1} = \{\mathcal{M}_{t-1} \setminus \{(c', s')\}\} \cup \{(c, s')\}$
11: set $a := (c', s)$ ▷ From Lemma 23, a will be a backward arc for loop recursion
12: **end while**
13: $\mathcal{M}_t = \mathcal{M}_{t-1} \cup \{a\}$ ▷ a now has no overlapping forward arcs, and is added to $\mathcal{M}_t$
14: **end if**
15: **end for**

▶ **Lemma 24.** *Suppose that at some time t, the algorithm adds a backward arc (c, s) to the matching $\mathcal{M}_t$ (either by rematching an older edge in step 10 during the recursion, or as a new edge in step 13) at the end of the recursion. Then, this arc does not overlap with any existing forward arcs in $\mathcal{M}_t$.*

Proof. The proof follows directly from the fact that we use the forward arc with the rightmost server in the step 9 of the algorithm. ◀

We now try to quantify the excess cost of RecursiveCancel over the optimal solution using the above-defined quantities. Indeed, if RecursiveCancel adds an edge (c, s) which is a backward arc at some time step t, then from Lemma 24, we know that it does not have any existing forward arc overlapping with it, and hence would not contribute to any sub-optimality at this time. Hence, RecursiveCancel is sub-optimal only due to the addition of forward arcs which have overlaps with existing backward arcs. These could happen either in step 5 or in step 10. However, by Lemma 23, we also know that the forward arcs added in step 10 are only suffixes of the original forward arcs, and so the excess cost due to the rematched forward arc is only at most that of the original forward arc. Using this intuition, we label each atomic interval of each forward arc as either *redundant* (i.e., cost avoided by OPT_t) or *non-redundant* (cost incurred by OPT_t) as follows:

▶ **Definition 25** (Redundant/non-redundant forward arcs with respect to atomic intervals)**.** *Suppose at time t, client c_t is matched to server s_t with a forward arc $a := (c_t, s_t)$ in step 5 of Algorithm 3. Then, the forward arc a is said to be redundant with respect to $I \in [\ell(c_t), \ell(s_t)]$ if $n^b_{t-1}(I) > n^f_{t-1}(I)$, and* non-redundant *with respect to I otherwise. Alternatively, if a new forward arc (c, s') is added in step 10 of the recursion then it must be the suffix of some forward arc $(c', s') \in \mathcal{M}_{t-1}$ (Lemma 23), and for any atomic interval I (or its subinterval), the new arc simply inherits its status (redundant/non-redundant) from the status of (c', s') with respect to I.*

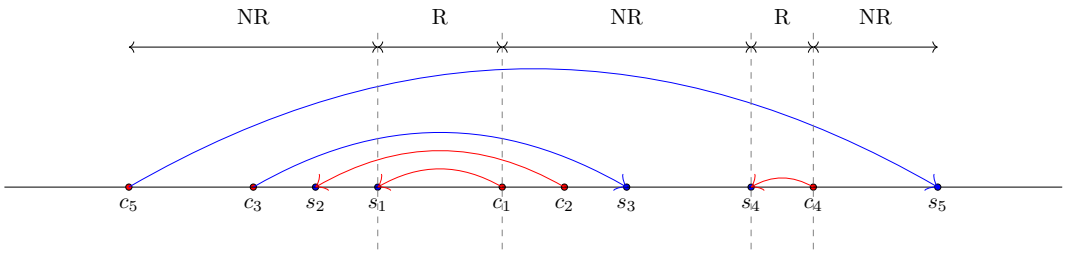


Figure 3 Illustration of Definition 25. In this example, when forward arc $a = (c_5, s_5)$ is added, a is redundant w.r.t. atomic intervals in $[\ell(s_1), \ell(c_1)]$ and $[\ell(s_4), \ell(c_4)]$ and *non-redundant* with respect to others.

We are now ready to bound the cost of RecursiveCancel. Indeed, we first show in Lemma 26 that the semantic meaning of redundant or non-redundant forward arcs with respect to atomic intervals is preserved throughout the algorithm. That is, for any atomic interval I, it will hold that there are exactly $\min(n_t^f(I), n_t^b(I))$ many forward arcs that are redundant with respect to I, since these can be avoided in an optimal matching.

▶ **Lemma 26.** *At any time t and for any atomic interval I, there are exactly* $\min(n_t^f(I), n_t^b(I))$ *forward arcs crossing I that are redundant w.r.t I.*

Proof. We prove the following two claims inductively on the number of client-server pairs added:

1. In any atomic interval $I \in \mathcal{AI}_t$, the number of forward arcs that are redundant with respect to I is equal to the minimum of the number of forward arcs and the number of backward arcs crossing I.
2. In any atomic interval I, if there exist two forward arcs $a_1 = (c_1, s_1)$ and $a_2 = (c_2, s_2)$ crossing the interval I such that a_1 is redundant w.r.t. I, and a_2 is non-redundant w.r.t. I, then $\ell(s_2) \geq \ell(s_1)$.

Let $a = (c, s)$ be the client-server pair given by Permutation. Consider the two cases, adding a forward arc and a backward arc:

1. Suppose that s is to the right of c i.e. the case when we add the forward arc directly. If in an atomic interval I between c and s, there are fewer forward arcs than backward arcs before adding c and s, then a is redundant w.r.t. I. Observe that this ensures that in such an interval I, the number of forward arcs that are redundant w.r.t. I increases, and is still equal to the minimum of the number of forward and the number of backward arcs crossing the interval. In intervals I where the number of forward arcs crossing I is at least the number of backward arcs crossing I before adding c and s, a is non-redundant w.r.t. I. In this case, the minimum of forward and backward arcs does not increase, and thus the claim continues to remain valid.
   For the second claim: If a is redundant w.r.t. an atomic interval I, then using claim 1 on the instance before adding a, we can infer that all the forward arcs crossing I are redundant w.r.t I. Thus, claim 2 is void in this case. If a is non-redundant w.r.t. an atomic interval I, we need to show that the new server is to the right of any server whose forward arc is redundant w.r.t. I. This follows directly from the fact that an unmatched server cannot be present in the middle of an arc (Lemma 14), and hence, s is to the right of the server of any forward arc that intersects I.
2. Suppose that s is to the left of c i.e. the case when we recursively add backward arc(s). In this case, in an atomic interval $I \in [\ell(s), \ell(c)]$, either a backward arc is added if there is no forward arc crossing I, or if there is at least one forward arc crossing I, the number of

forward arcs crossing I reduces by one. The intervals outside $[\ell(s), \ell(c)]$ are not affected. From Lemma 23, we know that the forward arcs corresponding to a server only shorten. Thus, the second claim trivially follows.

The algorithm deletes a forward arc from the atomic interval $I \in [\ell(s), \ell(c)]$ if there exists at least one forward arc crossing I before adding the new client c. If the number of forward arcs crossing I is at most the number of backward arcs crossing I, then all the forward arcs crossing I are marked redundant w.r.t. I, and we delete one such arc interval. The property of claim 1 still holds. Similarly, the property holds if there are no backward arcs are crossing I in which case, all the forward arcs crossing I are marked non-redundant. Thus, it remains to show that if there are both forward arcs that are labeled non-redundant and also ones that are marked redundant w.r.t. I cross I, our algorithm makes sure after the changes, one of the forward arc that is labeled non-redundant no longer crosses I. We use claim 2 here. If there are forward arcs a_1 and a_2 cross I and a_1 is redundant w.r.t I while a_1 is non-redundant w.r.t I, note that the server of a_2 is to the right of the server of a_1.

Recall that for any atomic interval I, there is at most one forward arc that crosses I is deleted. If a forward arc a that is redundant w.r.t. I is deleted, when the arc is selected in Line 9 of the algorithm, it has the farthest server among all arcs that intersect the backward arc. This combined with the above fact implies that if a forward arc that is redundant w.r.t. I is deleted, then there is no forward arc that is marked non-redundant crosses I. Thus, if there are forward arcs that are labeled non-redundant crossing I, our algorithm deletes one of them, which completes the proof of claim 1 in the case when we add a backward arc.

The both cases together complete the proof of the two claims, and in particular, of the original lemma. ◀

In order to analyze the cost of the algorithm, we define the *redundant cost* of a forward arc $a = (c, s)$ at time t as the sum of the lengths of all the atomic intervals $I \in \mathcal{AI}_t$ such that a is redundant w.r.t. I. Similarly, the *non-redundant cost* of a forward arc $a = (c, s)$ is the sum of the lengths of all the atomic intervals $I \in \mathcal{AI}_t$ such that a is non-redundant w.r.t. I. In Lemma 27, we bound the total redundant cost of all the forward arcs in terms of their non-redundant cost.

▶ **Lemma 27.** *At any time t, the redundant cost of any forward arc is at most the non-redundant cost of that forward arc.*

Proof. We will prove that for any forward arc a, at any time t, the redundant cost of a is at most the non-redundant cost of a at time t. Summing over all forward arcs will then complete the proof of Lemma 27.

To show this, we will in fact show the following: if RecursiveCancel adds a forward arc $a = (c_t, s_t)$ in step 5 at time t, then in *any suffix of* a, the redundant cost is at most the non-redundant cost. This suffices for us since we know from Lemma 23 that re-matched forward arcs are only suffixes of existing forward arcs, and their redundant/non-redundant labels remain the same from Definition 25.

Henceforth, we assume that at time t, the client c_t has arrived, and Permutation has chosen to match it using the server s_t by a forward arc, and RecursiveCancel has done the same. Let $A = C_{t-1} \cup S_{t-1}$ denote the set of clients and servers prior to adding c_t and s_t. Let $x \in [\ell(c_t), \ell(s_t)]$. Consider the suffix $[x, \ell(s_t)]$ of the arc (c_t, s_t), and introduce a *virtual client* c' at x. We claim that the optimal cost of matching clients and servers in $A \cup \{c', s_t\}$ is at least the optimal cost of matching clients and servers in A. Indeed, if this is

not the case, then we can produce a cheaper matching for the clients in C_{t-1} using a subset of servers in $S_{t-1} \cup \{s_t\}$ by ignoring the client c', which is a contradiction to Proposition 9 that Permutation maintains a solution that is server-optimal.

When we add $\{c', s_t\}$ to A, by Lemma 19, the increase in the cost of optimal matching occurs precisely at the intervals where the number of clients to the left is greater than the number of servers (after adding c' and s_t). And in other intervals in $[x, \ell(s_t)]$, the cost paid by the optimal matching decreases. However, this exactly corresponds to the redundancy and non-redundancy of the forward arc (c_t, s_t) with respect to the intervals inside $[x, \ell(s_t)]$. The intervals where the cost of optimal matching increases are the ones with respect to which the arc is non-redundant, and the intervals where the cost of optimal matching decreases are the ones with respect to which the arc is redundant. ◀

▶ **Theorem 28.** *At any point of time, the cost of the matching of Algorithm 3 (*RecursiveCancel*) is at most* 3 *times the cost of the optimal offline matching* OPT_t.

Proof. For the sake of analysis, for every atomic interval $I \in \mathcal{AI}_t$, let us label an arbitrary set of $\min(n_t^f(I), n_t^b(I))$ backward arcs as redundant with respect to I, and the rest as non-redundant with respect to I. Then, from Lemma 26, there will be an equal number of redundant backward arcs and redundant forward arcs with respect to any atomic interval $I \in \mathcal{AI}_t$. Using this, we conclude that, for any atomic interval $I \in \mathcal{AI}_t$, there are exactly $|\mathrm{disc}_t(I)|$ non-redundant arcs (including both forward and backward) w.r.t I. Indeed, if $n_t^f(I) \geq n_t^b(I)$, then there are $n_t^b(I)$ redundant forward (and redundant backward) arcs by Lemma 26, and the remaining $n_t^f(I) - n_t^b(I)$ forward arcs crossing I are non-redundant w.r.t I. But this is precisely the (absolute value of) the $\mathrm{disc}_t(I)$ by Claim 20. A similar argument holds when $n_t^b(I) > n_t^f(I)$.

For ease of notation, let us denote the total non-redundant cost of all the forward arcs (resp. backward) maintained by the algorithm at time t as $\mathsf{cost}(\mathcal{M}_t, NF)$ (resp. $\mathsf{cost}(\mathcal{M}_t, NB)$). Similarly, we denote the total redundant cost of all the forward arcs (resp. backward) as $\mathsf{cost}(\mathcal{M}_t, RF)$ (resp. $\mathsf{cost}(\mathcal{M}_t, RB)$). Now, from Lemma 19, and by noting that $|\mathrm{disc}_t(I)|$ is equal to the number of arcs crossing I which are non-redundant w.r.t. I, we have that

$$\mathsf{cost}(\mathcal{M}_t^*) = \sum_I |I||\mathrm{disc}_t(I)| = \mathsf{cost}(\mathcal{M}_t, NF) + \mathsf{cost}(\mathcal{M}_t, NB)\,.$$

On the other hand, the cost of $\mathcal{M}_t$ maintained by RecursiveCancel is at most

$$\begin{aligned}
\mathsf{cost}(\mathcal{M}_t) &= \mathsf{cost}(M_t, RF) + \mathsf{cost}(M_t, RB) + \mathsf{cost}(M_t, NF) + \mathsf{cost}(M_t, NB) \\
&= 2 \cdot \mathsf{cost}(M_t, RF) + \mathsf{cost}(M_t, NF) + \mathsf{cost}(M_t, NB) \\
&\leq 2 \cdot \mathsf{cost}(M_t, NF) + \mathsf{cost}(M_t, NF) + \mathsf{cost}(M_t, NB) \\
&\leq 3\,(\mathsf{cost}(\mathcal{M}_t, NF) + \mathsf{cost}(\mathcal{M}_t, NB)) \leq 3\mathsf{cost}(\mathcal{M}_t^*) = 3\mathsf{OPT}_t\,.
\end{aligned}$$

The first equality is from the definition of redundant backward arc w.r.t. atomic intervals and the first inequality is due to Lemma 27. ◀

4.3 Algorithm MinimumCancel

Even though the RecursiveCancel algorithm has a good competitive ratio, there are instances in which the algorithm performs $\Omega(k)$ rematches per client on average. We illustrate one such example in Appendix A. We, therefore, present another algorithm MinimumCancel which also satisfies Definition 21 and then bound its recourse.

Algorithm 4 Algorithm MinimumCancel.

1: **for** each client c_t arriving at time t **do**
2: let s_t be the server Permutation matches c_t to
3: **if** (c_t, s_t) is a forward arc **then**
4: $\mathcal{M}_t = \mathcal{M}_{t-1} \cup (c_t, s_t)$
5: **else**
6: let A_t denote all forward arcs $\overrightarrow{c,s} \in \mathcal{M}_{t-1}$ with $\ell(c) \in [\ell(s_t), \ell(c_t)]$
7: let $F_t = \cup_{\overrightarrow{c,s} \in A_t} [\ell(c), \ell(s)]$ denote the union of line segments contained within A_t, and let $O_t = F_t \cap [\ell(s_t), \ell(c_t)]$ denote the *overlapped* portion with $\overleftarrow{s_t, c_t}$
8: let A_t^* denote a *minimal subset* of A_t whose union covers the overlapped region O_t
9: order the edges in A_t^* as $(c'_1, s'_1), (c'_2, s'_2), \ldots, (c'_m, s'_m)$ such that $\ell(c'_1) \leq \ell(c'_2) \ldots \leq \ell(c'_m)$
10: let $\widetilde{A}_t = \{(c'_2, s'_1), (c'_3, s'_2), \ldots, (c'_m, s'_{m-1})\}$
11: update $\mathcal{M}_t = \{\mathcal{M}_{t-1} \setminus A_t^*\} \cup \widetilde{A}_t \cup \{(c_t, s'_m)\} \cup \{(c'_1, s_t)\}$
12: **end if**
13: **end for**

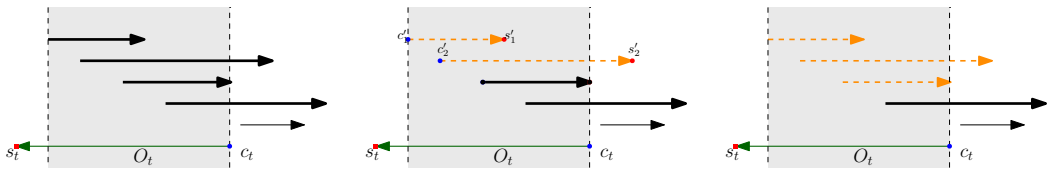
The crux of the algorithm is to cancel parts of forward arcs that overlap with the new backward arc in a manner that *minimally changes the existing solution*. We do this by identifying a minimal arc cover of the overlapping regions and perform the rematch only within the covering set of arcs. Figure 4a gives an illustration of the existing set of arcs when a new backward arc $\overleftarrow{s_t, c_t}$ is added. Figure 4b shows the minimal cover chosen in step 8 of the algorithm, and the final rematch will contain the arcs $\overleftarrow{s_t, c'_1}, \overrightarrow{c'_2, s'_1}$ and $\overrightarrow{c_t, s'_2}$. Figure 4c shows an example of a non-minimal cover where there are three overlapping forward arcs.

▶ **Lemma 29.** *When a new client c_t arrives, the re-matched set of forward arcs $\widetilde{A}_t$ computed in step 10 of Algorithm 4 are mutually disjoint.*

Proof. We first claim that in the minimum cover A_t^*, no three forward arcs intersect. Suppose for contradiction that there are forward arcs $a_1 = \overrightarrow{c_1, s_1}, a_2 = \overrightarrow{c_2, s_2}, a_3 = \overrightarrow{c_3, s_3}$ intersect at a point. Without loss of generality, let a_1 be the arc with left most client and a_3 be the arc with right most server. Then, the union of $[\ell(c_1, \ell(s_1))]$ and $[\ell(c_3), \ell(s_3)]$ fully contains $[\ell(c_2), \ell(s_2)]$, thus making the arc a_2 redundant in the cover, contradicting the minimality of the cover A_t^*. It is easy to conclude that the arcs $(c'_2, s'_1), (c'_3, s'_2), \ldots, (c'_m, s'_{m-1})$ are mutually disjoint. The minimal cover property is crucial as illutrated in Figure 4c; the disjointness property of residual forward arcs does not hold for the non-minimal cover. ◀

Using Lemma 29, we can bound the recourse of the MinimumCancel algorithm.

▶ **Theorem 30.** *After the arrival of k clients, the total recourse of* MinimumCancel *algorithm is at most $\mathcal{O}(k \log(k))$.*



(a) A backward arc and overlapping forward arcs.

(b) A minimal cover of O_t (in dashed orange).

(c) A non-minimal cover of O_t (in dashed orange).

Figure 4 Illustration for Algorithm 4. Forward arcs in A_t are shown with thick width.

Proof. Note that once a client is matched by a backward arc, it is not going to get re-matched later. We now bound the number forward-arc to forward-arc rematches.

For a vertex (either client or server) z, let us define a "length" $\text{len}(z)$ parameter which is equal to the number of vertices (all vertices which are part of the eventual matching after all the k clients have arrived) lying strictly inside the arc defined by z and its currently matched server/client. Therefore, before the start of the algorithm, the len value of every vertex is at most $2k$. We now define the level of vertex z as $\lfloor \log \text{len}(z) \rfloor$ so that the total initial level of all the vertices is at most $2k(1 + \log k)$. Suppose that a client c is currently matched to s by using a forward arc, and in an iteration gets re-matched to s' and s gets re-matched to c' both again using forward arcs. At least one of $\text{len}(c)$ or $\text{len}(s)$ should have decreased by at least a factor of 2 since these are now disjoint arcs from Lemma 29. And therefore the total level of c and s at least decreases by 1. In other words, on every re-match, the total level decreases by at least 1, which together with the bound on the total initial level gives the number of such re-matches to be at most $2k(1 + \log k)$. If at least one of c or s gets re-matched by a backward arc, its match does not change from then on. Thus, the number of these type of re-matches are at most $2k$. Thus, in total, the recourse of the algorithm is $\mathcal{O}(k \log k)$. ◀

5 Conclusion and Open Questions

The current work (together with the concurrent work [21]) represents the first attempt at exploring the trade-off between recourse and competitive ratio in online metric matching, and understanding the optimal trade-offs is an interesting direction to pursue. Concretely, can we get $o(\log k)$-competitive (even randomized) algorithms with $\text{polylog}(k)$-recourse for general metrics? This would be interesting given the $\Omega(\log k)$-lower bounds for algorithms without recourse. Similarly, obtaining or refuting $O(1)$-competitive algorithms with $O(1)$-recourse on line metrics is a very interesting question. Finally, extending our results to specialized inputs such as random order arrivals or unknown *i.i.d.* models would be interesting, as it better captures beyond-worst-case scenarios.

References

1. Antonios Antoniadis, Carsten Fischer, and Andreas Tönnis. A collection of lower bounds for online matching on the line. In *Latin American Symposium on Theoretical Informatics*, pages 52–65. Springer, 2018.
2. 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)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.
3. Yossi Azar and Amit Jacob Fanani. Deterministic min-cost matching with delays. In *International Workshop on Approximation and Online Algorithms*, pages 21–35. Springer, 2018.
4. Nikhil Bansal, Niv Buchbinder, Anupam Gupta, and Joseph Seffi Naor. An $O(\log^2 k)$-competitive algorithm for metric bipartite matching. In *European Symposium on Algorithms*, pages 522–533. Springer, 2007.
5. Marcin Bienkowski, Artur Kraska, and Paweł Schmidt. A match in time saves nine: Deterministic online matching with delays. In *International Workshop on Approximation and Online Algorithms*, pages 132–146. Springer, 2017.
6. Brian Brubach, Karthik Abinav Sankararaman, Aravind Srinivasan, and Pan Xu. New algorithms, better bounds, and a novel model for online stochastic matching. In *24th Annual European Symposium on Algorithms (ESA 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

7 Nikhil R Devanur, Balasubramanian Sivan, and Yossi Azar. Asymptotically optimal algorithm for stochastic adwords. In *Proceedings of the 13th ACM Conference on Electronic Commerce*, pages 388–404. ACM, 2012.

8 Yuval Emek, Shay Kutten, and Roger Wattenhofer. Online matching: haste makes waste! In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 333–344. ACM, 2016.

9 Björn Feldkord, Matthias Feldotto, Anupam Gupta, Guru Guruganesh, Amit Kumar, Sören Riechers, and David Wajc. Fully-dynamic bin packing with little repacking. In *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018, July 9-13, 2018, Prague, Czech Republic*, pages 51:1–51:24, 2018.

10 Bernhard Fuchs, Winfried Hochstättler, and Walter Kern. Online matching on a line. *Theoretical Computer Science*, 332(1-3):251–264, 2005.

11 Gagan Goel and Aranyak Mehta. Online budgeted matching in random input models with applications to adwords. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 982–991. Society for Industrial and Applied Mathematics, 2008.

12 Anupam Gupta, Guru Guruganesh, Binghui Peng, and David Wajc. Stochastic online metric matching. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 67:1–67:14, 2019.

13 Anupam Gupta, Ravishankar Krishnaswamy, Amit Kumar, and Debmalya Panigrahi. Online and dynamic algorithms for set cover. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 537–550, 2017.

14 Anupam Gupta, Amit Kumar, and Cliff Stein. Maintaining assignments online: Matching, scheduling, and flows. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 468–479, 2014.

15 Anupam Gupta and Kevin Lewi. The online metric matching problem for doubling metrics. In *International Colloquium on Automata, Languages, and Programming*, pages 424–435. Springer, 2012.

16 Varun Gupta, Ravishankar Krishnaswamy, and Sai Sandeep. PERMUTATION strikes back: The power of recourse in online metric matching. *arXiv preprint*, 2019. `arXiv:1911.12778`.

17 Bala Kalyanasundaram and Kirk Pruhs. Online weighted matching. *J. Algorithms*, 14(3):478–488, 1993. `doi:10.1006/jagm.1993.1026`.

18 Samir Khuller, Stephen G Mitchell, and Vijay V Vazirani. On-line algorithms for weighted bipartite matching and stable marriages. *Theoretical Computer Science*, 127(2):255–267, 1994.

19 Elias Koutsoupias and Akash Nanavati. The online matching problem on a line. In *International Workshop on Approximation and Online Algorithms*, pages 179–191. Springer, 2003.

20 Jannik Matuschke, Ulrike Schmidt-Kraepelin, and José Verschae. Maintaining perfect matchings at low cost. In *46th International Colloquium on Automata, Languages, and Programming, ICALP 2019, July 9-12, 2019, Patras, Greece.*, pages 82:1–82:14, 2019.

21 Nicole Megow and Lukas Nölke. Online minimum cost matching on the line with recourse. *arXiv preprint*, 2020. `arXiv:2001.03107`.

22 Adam Meyerson, Akash Nanavati, and Laura Poplawski. Randomized online algorithms for minimum metric bipartite matching. In *Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 954–959. Society for Industrial and Applied Mathematics, 2006.

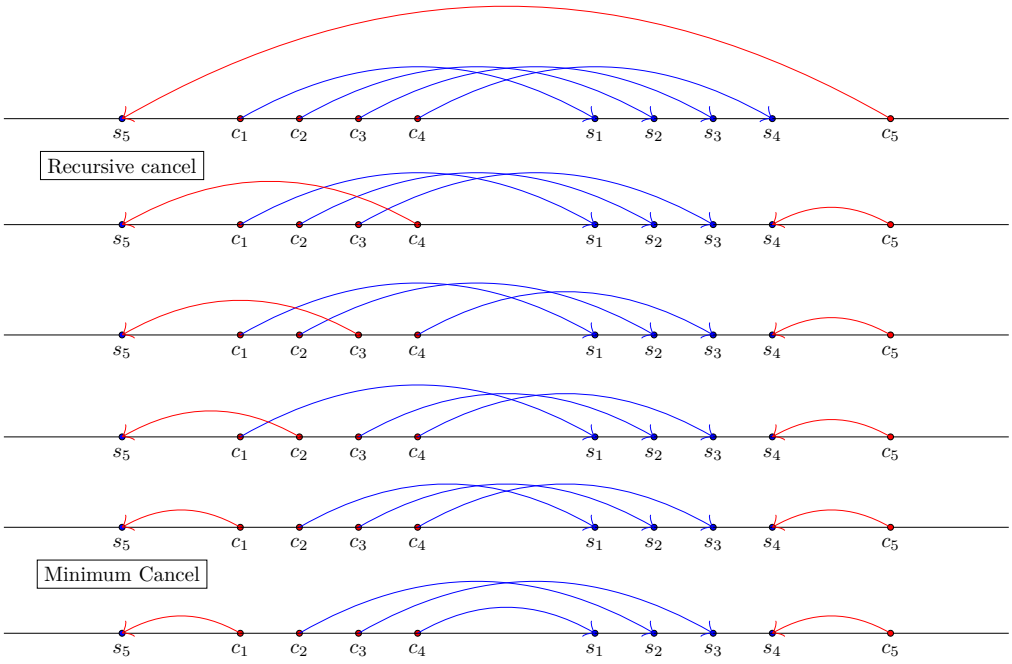
23 Krati Nayyar and Sharath Raghvendra. An input sensitive online algorithm for the metric bipartite matching problem. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 505–515. IEEE, 2017.

24 Sharath Raghvendra. A robust and optimal online algorithm for minimum metric bipartite matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

25 Sharath Raghvendra. Optimal analysis of an online algorithm for the bipartite matching problem on a line. In *34th International Symposium on Computational Geometry, SoCG 2018, June 11-14, 2018, Budapest, Hungary*, pages 67:1–67:14, 2018.

A Bad Example for Recourse of RecursiveCancel

We illustrate the fact that RecursiveCancel algorithm can have bad recourse in the following example:



In this instance, there are four clients c_1, c_2, c_3 and c_4, $\ell(c_1) < \ell(c_2) < \ell(c_3) < \ell(c_4)$ currently matched using forward arcs to s_1, s_2, s_3 and s_4 respectively such that $\ell(c_4) < \ell(s_1) < \ell(s_2) < \ell(s_3) < \ell(s_4)$. A new client c_5 arrives to the right of s_4 and Permutation outputs s_5 to the left of c_1 as the new server. Since the arc (c_5, s_5) is backward, the algorithms try to fix the matching. The RecursiveCancel i.e. Algorithm 3 changes the matching completely to obtain $(c_2, s_1), (c_3, s_2), (c_4, s_3)$ as the new forward arcs, where as Algorithm 4 changes only c_4's matching and keeps c_2 and c_3 intact. If there are k such forward arcs, and if k backward arcs arrive, Algorithm 3 has a recourse of $\Omega(k^2)$ where as Algorithm 4 has only $O(k)$ recourse.

B Missing Proofs

Proof of Proposition 12. For the sake of simplicity, we stick with $d = 2$, and the same proof holds for larger d as well. At any time t, we view our algorithm as simulating the Permutation algorithm for a certain batch sequence. Indeed, note, the solution maintained in $\mathcal{M}_t$ is exactly what Permutation maintains when fed $O(\log t)$ batches of consecutive clients corresponding to the different powers-of-two 2^{i-1} (in decreasing order) such that the

i^{th} bit from right in the binary representation of t is 1. Theorem 11 then bounds the cost. The recourse is bounded since any client is involved in a re-matching of size 2^i at most once for all i. ◀

Proof of Proposition 13. For simplicity, we prove the proposition for the case $d = 2$. An alternate view of Algorithm MultiScalePermutation is the following: Let the leaves of a complete balanced tree of degree d denote the k client arrivals. Whenever the arrival of a client completes a subtree (that is, it is the rightmost leaf in some subtree), the matching for the clients and their currently matched serves is re-solved optimally.

Our lower bound instance will be on the line metric and consist of two parts: a *core* instance and a *auxiliary* instance. The subsequent client arrival will be chosen as the next arrival from either the core or the auxiliary instance so as to obtain a large recourse cost as we describe soon. The servers for the core instance will at locations $\pm 1, \pm 2, \pm 3, \ldots, \pm k/2$, and the servers for the auxiliary instance will be k servers at location $10k$. The client arrival sequence in the core instance will be $\epsilon, -1-\epsilon, 1+\epsilon, -2-\epsilon, 2+\epsilon, -3-\epsilon, 3+\epsilon, \ldots$; the client arrival sequence for the auxiliary instance is $10k, 10k, 10k, \ldots$.

Note that the above instance have been set up so that on the arrival of a client from the core instance, the server added by Permutation to the matching is also from the core instance, and similarly for a client from the auxiliary instance a server from the auxiliary instance is added. Further, the same is done by OPTso that it suffices to study the cost and recourse for the arrivals in the core instance.

To decide whether the next client arrival happens from the core or the auxiliary sequence, we first check whether the arrival completes any subtree. If it does, denote the largest subtree it completes by T, and by $T_1, \ldots, T_d$ the d subtrees of the root of T (so that the new arrival is the rightmost leaf of T_d). If the number of core arrivals so far in T_d is even, then the new arrival is also chosen from the core sequence. Otherwise the new arrival is chosen from the auxiliary chosen.

We first prove the recourse bound. The sequence in which Permutation adds servers when the clients arrive from the core sequence is $1, -1, 2, -2, \ldots$. In particular, the new client and server are added on the opposite sides of a central matching that is built online. The construction of the client arrival sequence ensures that when MultiScalePermutation resolves the optimal matching for subtree $T = (T_1, T_2)$ the number of client arrivals in T_2 is odd, and hence batch resolving ends up rematching all clients in $T_1 \cup T_2$ (except at most one). (In the general d case we have to assume d is odd, in which case it is easy to show that all the subtrees $T_1, T_2, \ldots, T_d$ have odd number of core clients, and thus a $\frac{d-1}{d}$ fraction of clients are rematched in the subtree T.)

To study cost, consider the matching immediately after the arrival of the ith client, and let $i = \sum_{j=0}^{\ell} d^j k_j$ $(0 \le k_j \le d-1)$ denote the base d representation of i. In particular, consider the case $k_j = 1$ for $1 \le j \le \ell$. The matching consists of one batch of d^ℓ clients each, followed by 1 batch of $d^{\ell-1}$ clients and so forth. The cost of OPTis at most i. However, the cost of MultiScalePermutation is $\Omega(i \cdot \ell) = \Omega(i \log_d i)$. ◀

Proof of Lemma 14. Recall that Permutation maintains an offline optimal matching $\mathcal{M}_t^*$ at time t, and when a client c_t arrives, we pair it with the server that is present in $\mathcal{M}_t^* \setminus \mathcal{M}_{t-1}^*$. In fact, the symmetric difference of $\mathcal{M}_t^*$ and $\mathcal{M}_{t-1}^*$ is an augmenting path starting at c_t and ending at s_t. Let it be denoted by $P = c_t, s_{p_1}, c_{p_1}, \ldots, s_{p_m} = s_t$. The edges $(c_t, s_{p_1}), (c_{p_1}, s_{p_2}), \ldots, (c_{p_{m-1}}, s_{p_m})$ are the new edges, and the rest $(s_{p_1}, c_{p_1}), (s_{p_2}, c_{p_2}), \ldots,$ $(s_{p_{m-1}}, c_{p_{m-1}})$ are the old edges. Recall that the cost of $\mathcal{M}_t^*$ is at least that of $\mathcal{M}_{t-1}^*$, and thus, in the augmenting path, the cost of new edges is at least that of the old edges.

We claim a stronger property that in any suffix of the augmenting path, the cost of the new edges is at least that of old edges. Consider a suffix $s_{p_i}, c_{p_i}, \ldots, s_{p_m}$. If the cost of new edges is less than the old edges, we can change the old matching from $(s_{p_i}, c_{p_i}), \ldots, (s_{p_{m-1}}, c_{p_{m-1}})$ to $(c_{p_i}, s_{p_{i+1}}), \ldots, (c_{p_{m-1}}, s_{p_m})$ while keeping the rest of the edges intact to get a matching with cost less than $\mathcal{M}^*_{t-1}$, contradicting the fact that $\mathcal{M}^*_{t-1}$ is an optimal matching for the first $t-1$ clients. Now, suppose that there is a free server s' in between c_t and s_t. Consider the prefix of the augmenting path P starting at c_t and ending at s'. Let this prefix be denoted by P'. Let $\mathcal{M}'_t$ be the matching obtained from $\mathcal{M}^*_{t-1}$ by augmenting with the new augmenting path P'. Since the cost of new edges is at least that of old edges in any suffix of the original augmenting path, the difference between new edges and old edges in P' is at most that of the original augmenting path P. Thus, the cost of the matching $\mathcal{M}'_t$ is at most that of $\mathcal{M}^*_t$. Furthermore, as we have assumed that the location of all the clients and servers are distinct, the cost of $\mathcal{M}'_t$ is strictly smaller than $\mathcal{M}^*_t$, contradicting the fact that $\mathcal{M}^*_t$ is an optimal matching of first t clients. This proves the claim that when we execute Permutation on the line metric, there are no free servers inside any arc. ◀

C Lower bounds

In this section, we present our lower bound for OMM on general metrics.

▶ **Theorem 31.** *Suppose that there exists an algorithm for* OMM *such that for every client c, the number of servers s such that c is matched to s at some point of execution of the algorithm is at most C, for an absolute constant C. Then, the competitive ratio of the algorithm is at least $\Omega(\log(n))$.*

Proof. We first describe the hard instance for OMM that we use to prove the lower bound. The underlying metric space is the star metric i.e. there exists a node v_0 that is at the center of the star, and a set of nodes $v_1, v_2, \ldots, v_n$ such that $d(v_0, v_i) = 1$ for all $i \in [n]$, and $d(v_i, v_j) = 2$ for all $i, j \in [n], i \neq j$. For every $i \in [n]$, there is a server s_i at v_i. For each time $t = 0, 1, \ldots, n-1$ a single client c_t arrives at a point in the metric space. First, at $t = 0$, the client c_0 arrives at v_0. The next clients arrive at the location of the server just used by the algorithm. Suppose that the algorithm matches c_0 to s_i. Then, c_1 arrives at v_i. After t clients have arrived and have been matched by the algorithm, consider the server matched to c_0- let it be s_{i_1}. Let s_{i_2} be the server matched by the algorithm to the client at v_{i_1}, and so on till there is no client yet arrived at v_{i_k}. Then, in our instance, at time t, a new client arrives at v_{i_k}.

Note that all the clients arrive at different locations in the metric space. This implies that at any point of time t, the offline optimal algorithm cost is equal to 1. We can simply match each client c_i other than c_0 to the server s_i, and match c_0 to an arbitrary unused server.

Suppose that for each client c, the number of servers s such that (c, s) is part of the matching of the algorithm at some point, is at most C. Then, we claim that there is a time t such that the online algorithm has cost at least $\Omega(\log(n))$ at time t. Let M_t, $t = 0, 1, \ldots, n-1$ denote the matching maintained by the algorithm after time t. We consider a new algorithm that maintains a set of matchings M'_t, $t = 0, 1, \ldots, n-1$ after time t. For every time t, we obtain M'_t from M_t as follows: Let $M = M_t$. While there exists a client c located at v_i matched in M to a server s at $v_j \neq v_i$, but the server s_i is not used in M, we rematch c to s_i in M. Note that this process terminates in at most n steps. When this process can no longer proceed, we output $M'_t = M$. The cost of the matching M'_t is at most the cost of M_t, as every iteration of the above procedure only decreases the cost of the matching. For every

client c, the number of servers s such that c is matched to s in some M'_t, is at most $C+1$. Finally, the new algorithm that maintains the matchings M'_t has a key property that at any time t: the matching M'_t can be described as a path: $(c_0, s_{i_1}), (c_1, s_{i_2}), \ldots, (c_t, s_{i_{t+1}})$ such that c_j and s_{i_j} are at the same location.

We now claim that there exist some time t such that the size of M'_t is at least $\Omega(\log(n))$, which proves the required lower bound. We define a directed graph $G = (V, E)$. The vertex set of the graph V is equal to $\{0, 1, \ldots, n\}$. There is an edge from i to j if for some time t, the client c_i is matched to the server s_j in M'_t. The out-degree of every node is at most $C+1$ in G. We also define the graphs $G_0, G_1, \ldots, G_{n-1}$ as follows: The vertex set of G_k is the same as G for every k. There is an edge from i to j in G_k if client c_i is matched to s_j in M'_k. It follows from the definitions that for every $k \in \{0, 1, \ldots, n-1\}$, G_i is a subgraph of G.

Note that for each k, the graph G_k is a path that starts at 0 and ends at the index of the location of the client c_k. Thus, all the graphs $G_0, G_1, \ldots, G_{n-1}$ are different path subgraphs of G all of which start at vertex 0 and end at a different vertex in G. As the out-degree of every vertex is at most $C+1$ in G, the number of distinct paths of length at most l in G starting at 0 is at most $(C+1)^l$. Thus, there should exist at least one path whose length is $\frac{\log(n)}{\log(C+1)} = \Omega(\log(n))$. As the length of the subgraph G_i denotes the cost of the matching M'_i, we get the required lower bound on the competitive ratio. ◀

How to Cut a Ball Without Separating: Improved Approximations for Length Bounded Cut

Eden Chlamtáč
Ben Gurion University of the Negev, Beer Sheva, Israel
https://www.cs.bgu.ac.il/~chlamtac/
chlamtac@cs.bgu.ac.il

Petr Kolman
Charles University, Faculty of Mathematics and Physics, Prague, Czech Republic
https://kam.mff.cuni.cz/~kolman/
kolman@kam.mff.cuni.cz

Abstract

The Minimum Length Bounded Cut problem is a natural variant of Minimum Cut: given a graph, terminal nodes s, t and a parameter L, find a minimum cardinality set of nodes (other than s, t) whose removal ensures that the distance from s to t is greater than L. We focus on the approximability of the problem for bounded values of the parameter L.

The problem is solvable in polynomial time for $L \leq 4$ and NP-hard for $L \geq 5$. The best known algorithms have approximation factor $\lceil (L-1)/2 \rceil$. It is NP-hard to approximate the problem within a factor of 1.17175 and Unique Games hard to approximate it within $\Omega(L)$, for any $L \geq 5$. Moreover, for $L = 5$ the problem is $4/3 - \varepsilon$ Unique Games hard for any $\varepsilon > 0$.

Our first result matches the hardness for $L = 5$ with a 4/3-approximation algorithm for this case, improving over the previous 2-approximation. For 6-bounded cuts we give a 7/4-approximation, improving over the previous best 3-approximation. More generally, we achieve approximation ratios that always outperform the previous $\lceil (L-1)/2 \rceil$ guarantee for any (fixed) value of L, while for large values of L, we achieve a significantly better $((11/25)L + O(1))$-approximation.

All our algorithms apply in the weighted setting, in both directed and undirected graphs, as well as for edge-cuts, which easily reduce to the node-cut variant. Moreover, by rounding the natural linear programming relaxation, our algorithms also bound the corresponding bounded-length flow-cut gaps.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Approximation Algorithms, Length Bounded Cuts, Cut-Flow Duality, Rounding of Linear Programms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.41

Category APPROX

1 Introduction

In the Minimum Length Bounded Cut problem, we are given a directed (undirected) graph $G = (V, E)$, two distinguished vertices $s, t \in V$, called the source and the sink, and an integer parameter $L > 0$, and wish to find a minimum cardinality *L-bounded node cut* (edge cut, resp.): a subset $F \subseteq V \setminus \{s, t\}$ of vertices (a subset $F \subseteq E$ of edges, resp.) such that the every path between the source s and the sink t in $G \setminus F$ has length strictly greater than L; the path length is the number of edges in it.

Various aspects of the problem have been studied for almost half a century: its relationship to the maximum L-bounded flow [1, 2, 3, 17] and to the maximum number of disjoint L-bounded $s - t$ paths [1, 7, 16], its complexity [5, 3, 15] and approximability [3, 14], fixed-parameter tractability [6, 9, 10, 11], and polynomial time solvability for various graph classes [4, 9, 18, 14]. In this paper we focus on the approximability of the problem.


© Eden Chlamtáč and Petr Kolman;
licensed under Creative Commons License CC-BY

There are four basic versions of the L-Bounded Cut problem, depending on whether the graph G is directed or undirected, and whether we cut nodes or edges. As observed by Golovach and Thilikos [11], an α-approximation algorithm for Directed L-Bounded Node Cut yields an α-approximation algorithm for Undirected L-Bounded Node Cut, Undirected $(L-1)$-Bounded Edge cut, and Directed $(L-1)$-Bounded Edge Cut. Thus, we describe all our algorithms just for the directed node version, and in the rest of the paper we focus exclusively on this setting.

On the complexity and approximability side, the node (edge, resp.) version of Minimum L-Bounded Cut is solvable in polynomial time for $L \leq 4$ [16] (for $L \leq 3$ [17], resp.) and is NP-hard for $L \geq 5$ (for $L \geq 4$, resp.) [3]. With respect to the length parameter L, there are several simple $O(L)$-approximation algorithms [3, 17], and the best known algorithm has approximation ratio $\lceil (L-1)/2 \rceil$ [3].

For $L \geq 4$, L-Bounded Edge Cut is known to be NP-hard to approximate within a factor of 1.1377 [3]. In fact, we observe that the reduction in that paper also implies the NP-hardness of approximating the problem to within a factor of 1.1715, and Unique Games hardness of approximating it to within $4/3 - \varepsilon$ for any $\varepsilon > 0$ – see Appendix A for details. Note that these results imply the same hardness for L-Bounded Node Cut for $L \geq 5$. Recently, Lee [15] showed that for bounded values of L it is Unique Games hard to approximate the undirected edge variant within $\Omega(\sqrt{L})$, and the other three variants within $\Omega(L)$. Thus, assuming the Unique Games Conjecture, the best possible approximation for all but the undirected edge variant is $\Theta(L)$. However, the exact best possible approximation as a function of L is not known.

If the length bound is not a fixed constant but a part of the input, the gap between the known hardness results and approximations is even bigger: there are no stronger hardness results and the best known algorithms [3] have approximation ratio $O(\min\{L, n/L\}) \subseteq O(\sqrt{n})$ in the node case and $O(\min\{L, n^2/L^2, \sqrt{m}\}) \subseteq O(n^{2/3})$ in the edge case where m denotes the number of edges.

1.1 Our results

Our first result is an algorithmic upper bound matching the Unique Games hardness of 4-Bounded Edge Cut and 5-Bounded Node Cut; note that these two problems are the *first* hard instances of L-Bounded Cut – for L the corresponding problems are in P. For reasons explained earlier we state all our results for the node case only.

▶ **Theorem 1.** *There is a $\frac{4}{3}$-approximation algorithm for Minimum 5-Bounded Node Cut.*

Similarly, for $L = 6$ we also describe a new algorithm with an improved approximation ratio.

▶ **Theorem 2.** *There is a $\frac{7}{4}$-approximation algorithm for Minimum 6-Bounded Node Cut.*

This algorithm is based on the same, yet more involved, techniques as the algorithm for $L = 5$. More generally, we have an approximation algorithm that works for any value of L.

▶ **Theorem 3.** *For any fixed length bound $L \geq 6$, there exists an $\left(\frac{L-1}{2} - \frac{3}{L-2}\right)$-approximation algorithm for Minimum L-Bounded Node Cut.*

This algorithm is based on our algorithm for $L = 6$ and a general observation that an α-approximation algorithm for L-Bounded Cut with certain properties can be used to design an approximation algorithm for $(L+1)$-Bounded Cut, with a slightly weaker approximation ratio (see Theorem 10). This is always better than the previous best known $\lceil (L-1)/2 \rceil$-approximation, but for large values of L it is not significantly better. Though various algorithmic techniques, including the above theorem, all point to $L/2 - o(L)$ being the best possible approximation (cf. [3]), we are able to improve over this bound.

▶ **Theorem 4.** *For any fixed length bound $L > 5$, there exists an $((11/25)L + O(1))$-approximation algorithm for* Minimum L-Bounded Node Cut.

A succinct summary of the old and the new results on approximability of Minimum L-Bounded Cut is provided in Table 1. It is worth mentioning that all our algorithms work also in the more general setting where every node (edge) has a non-negative weight and the objective is to find an L-bounded cut of minimum total weight, simply by including these weights in the objective function of the linear program.

Table 1 Known and **new** (bold type) results for bounded values of L. All results hold for both directed and undirected cases, unless stated otherwise. We note that a hardness result for some value of L implies the same hardness result for larger values as well, thus we only state hardness results here for the smallest L for which they hold. NP stands for NP hardness of approximation, UG for Unique Games hardness of approximation.

L	Node Cuts		Edge Cuts	
	Hardness	Approximations	Hardness	Approximations
≤ 3	1 (in P)		1 (in P) [17]	
4	1 (in P) [16]		NP 1.1715 [3, 12] UG 4/3 [3, 13]	2 [3] **4/3**
5	NP 1.1715 [3, 12] UG 4/3 [3, 13]	2 [3] **4/3**		3 [3] **7/4**
6		3 [3] **7/4**		3 [3] **12/5**
any	UG $\Omega(L)$ [15]	$\lceil (L-1)/2 \rceil$ [3] $\mathbf{(L-1)/2 - 3/(L-2)}$ $\mathbf{0.44(L-1) + O(1)}$	UG $\Omega(L)$ directed, $\Omega(\sqrt{L})$ undir. [15]	$\lceil L/2 \rceil$ [3] $\mathbf{L/2 - 3/(L-1)}$ $\mathbf{0.44L + O(1)}$

Approximate duality of L-bounded flows and cuts

As we have already mentioned earlier, researchers investigated the relation between the minimum L-bounded cut and the maximum L-bounded flow (where an L-bounded flow is a flow that can be decomposed into flows along paths of length at most L) (e.g., [2, 3, 17]); in fact, this was one of the first questions Adámek and Koubek [1] asked back in 1971.

All our algorithms are based on rounding the linear program (1) (given in the next subsection) which is the dual of the exact linear programming formulation of the maximum L-bounded flow. Thus, as a corollary of our algorithmic results, we obtain improved approximate duality relations between L-bounded cuts and flows that are tighter than those previously known. For the sake of brevity we state the result just for 4-bounded edge cuts and flows.

▶ **Corollary 5.** *Given a graph $G = (V, E)$ and nodes $s, t \in V$, let $\mathcal{F}$ denote a maximum 4-bounded flow, and $\mathcal{C}$ a minimum 4-bounded edge cut. Then*

$$|\mathcal{F}| \leq |\mathcal{C}| \leq \frac{4}{3}|\mathcal{F}|,$$

and these bounds are tight.

1.2 Overview of our approach

All our algorithms are based on a rounding of a natural linear programming relaxation of the problem. This relaxation was studied in earlier works on L-Bounded Cut (e.g., [3, 17]) but for the sake of completeness we provide it here again. In the following linear program (solvable via a simple separation oracle), $\mathcal{P}_{s,t}(L)$ denotes the set of all paths between s and t of length at most L.

$$\min \sum_{v \in V \setminus \{s,t\}} x_v \tag{1}$$

$$\sum_{v \in p \setminus \{s,t\}} x_v \geq 1 \quad \forall p \in \mathcal{P}_{s,t}(L) \tag{2}$$

$$x_v \geq 0 \quad \forall v \in V$$

The previous approximation algorithms were based on cutting only shortest paths, which can be done optimally by taking a minimum cut in the layered graph of shortest paths from s to t. By iteratively cutting all shortest $s-t$ paths until the distance between s and t becomes larger then L, we get an L-approximation.

This approach can also be framed as a rounding of the above linear programming relaxation, via the following classical rounding which cuts in every iteration all shortest paths while paying at most the LP value:

Exact-Round$(G, (x_v)_{v \in V})$

- For every $v \in V \setminus \{s,t\}$, let $y_v := \min\{x(p) \mid p : s \rightsquigarrow v,\ |p| = d(s,v)\}$ and $I_v := [y_v, y_v + x_v]$, where $s \rightsquigarrow v$ stands for an $s-v$ path, d is the hop-distance in G and $x(p)$ is defined to be the total LP value of vertices in p excluding its endpoints.
- Sample $r \in [0,1]$ uniformly at random.
- Return the cut $\{v \mid r \in I(v)\}$.

In words, every vertex $v \in V \setminus \{s,t\}$ is mapped to an interval I_v of length x_v in such a way that for every $s-t$ path p of interest (i.e., path of length $d(s,v)$), we have $[0,1] \subseteq \bigcup_{v \in p} I_v$. The algorithm then cuts all vertices whose corresponding intervals lie on the boundary of a ball of uniformly random radius; by cutting these vertices the algorithm separates s from t in the layered subgraph of shortest $s-t$ paths, and, thus, increases the distance between s and t on the whole graph by one at least. The property that is important in the analysis, and that will be important later in our new algorithms, is the following:

▶ **Observation 6.** *Given a solution $(x_v)_{v \in V}$ of the LP (1), for every $v \in V$, **Exact-Round** cuts v with probability at most x_v.*

The difficulty with the outlined iterative approach to construct an L-bounded cut is that it uses up to L iterations (or up to $\lceil (L-1)/2 \rceil$, if we first cut all vertices with $x_v \geq 1/\lceil (L-1)/2 \rceil$), where each iteration may yield a cut as large as the LP value. See Figure 1 for an instance where more than one iteration is performed. Our idea is to circumvent the need for multiple iterations by mapping every vertex v to *multiple* intervals, each interval representing a possible position of the vertex in an L-bounded path. Thus, our algorithms are similar to classical ball cutting algorithms, yet they differ in that they do not (necessarily) separate s from t.

However, this is only a framework for our improvements. A naïve random ball growing algorithm would not yield a better approximation ratio when applied to these intervals. To fully take advantage of this framework, we introduce additional ideas such as cutting only a carefully chosen subset of the boundaries of more than one ball, and/or modifying the LP values in order to take better advantage of the structure of our mappings.

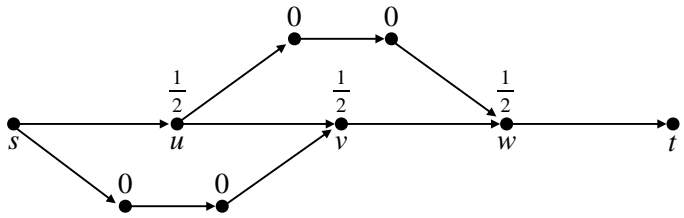


Figure 1 In this instance, $L = 5$, and the LP values appear above the vertices. In the first iteration of **Exact-Round**, we have $I_u = [0, 1/2]$, $I_v = [1/2, 1]$, and $I_w = [1, 3/2]$. Intervals for other vertices will have length 0. Thus, in this iteration either u or v will be cut (no other vertex is assigned an interval with a positive length intersection with $[0, 1]$), and we will need a second iteration to cut some remaining path of length 5.

2 A 4/3-approximation for $L = 5$

Here we prove Theorem 1. Specifically, we give a rounding of the natural LP relaxation for 5-Bounded Node Cut which cuts every vertex $v \in V$ with probability at most $(4/3) \cdot x_v$.

Before describing our rounding, we describe here a certain mapping of the vertices $V \setminus \{s, t\}$ to intervals. For a graph $G = (V, E)$ and an LP solution $(x_v)_{v \in V}$, we map every vertex v at distance most 2 from s or to t to two intervals, $I^+_G(v), I^-_G(v)$. For any vertices u, v such that $(s, u), (v, t) \in E$, we define

$$I^-_G(u) = I^+_G(u) = [0, x_u] \qquad I^-_G(v) = I^+_G(v) = [1 - x_v, 1].$$

Next, for any vertex u (for which $(u, t) \notin E$) at distance 2 from s, define

$$y^+_G(u) = \min_{u':(s,u'),(u',u) \in E} x_{u'} \qquad I^+_G(u) = [y^+_G(u), y^+_G(u) + x_u].$$

If $d_G(u, t) > 2$, define $I^-_G(u) = I^+_G(u)$; note that in such a case, within any 5-bounded $s - t$ path containing it, u will be at distance 2 from s and 3 from t. Finally, for any vertex v (for which $(s, v) \notin E$) at distance 2 to t, define

$$y^-_G(v) = \min_{u:(v,v'),(v',t) \in E} x_{v'} \qquad I^-_G(v) = [1 - y^-_G(v) - x_v, 1 - y^-_G(v)].$$

If $d_G(s, v) > 2$, define $I^+_G(v) = I^-_G(v)$.

We will drop the subscript G from the above definitions when it is clear from the context. Note that $I^+(v), I^-(v)$ have length x_v for every v, and $I^-(v) = I^+(v)$ for all vertices v except those which are both at distance 2 from s and at distance 2 to t. For v such that $I^-(v) \neq I^+(v)$, there exists a path $\langle s, u', v, v', t \rangle$ such that $y^+(v) = x_{u'}$ and $y^-(v) = x_{v'}$. Thus, the intervals have the same length, and $I^-(v)$ starts to the left of $I^+(v)$ (when plotted on the real number line). Indeed, the left endpoint of $I^+(v)$ will be $x_{u'}$, while the left endpoint of $I^-(v)$ will be $1 - (x_{v'} + x_v)$, which is at most $x_{u'}$ by Constraint (2).

We are now ready to define our rounding:

Algorithm 5-Round

- Let $C_0 = \{v \in V \mid x_v \geq 3/4\}$. Let G' be the remaining graph after deleting all vertices in C_0.
- Sample $r \in [0, 1]$ uniformly at random.

- Sample $r_1 \in [0, 1/2]$ uniformly at random, and let $r_2 = r_1 + 1/2$.
- Let C_1 be the set of all vertices v such that $r \in I^-_{G'}(v) \cup I^+_{G'}(v)$.
- Let C_2 be the set of all vertices v for which at least one of the following conditions holds:
  1. $r_1 \in I^-_{G'}(v) \cap I^+_{G'}(v)$ or $r_2 \in I^-_{G'}(v) \cap I^+_{G'}(v)$, or
  2. $r_1, r_2 \in I^-_{G'}(v)$ or $r_1, r_2 \in I^+_{G'}(v)$.
- With probability 2/3, return $C_0 \cup C_1$. Otherwise (with probability 1/3), return $C_0 \cup C_2$.

▶ Remark 7. This algorithm can be easily derandomized by trying $O(|V|)$ possible radii r, r_1 and choosing the better of the two cuts $C_0 \cup C_1$, $C_0 \cup C_2$.

First let us see why this is a valid 5-bounded cut.

▶ **Lemma 8.** *Algorithm **5-Round** cuts all paths of length at most 5 from s to t.*

Proof. Since vertices in C_0 are always cut, we focus on paths not cut by C_0. That is, on paths in G'.

First, let us see that C_1 cuts all 5-bounded $s-t$ paths in G' (the proof for paths of length < 5 is similar or simpler). Let $p = \langle s, u', u, v, v', t\rangle$ be such a path and assume that $(s,u), (v,t) \notin E$; otherwise cutting a vertex in a shorter path (i.e., $\langle s, u, v, v', t\rangle$ or $\langle s, u', u, v, t\rangle$) cuts p as well. Let u'', v'' be vertices such that $y^+(u) = x_{u''}$ and $y^-(v) = x_{v''}$. Since $\langle s, u'', u, v, v'', t\rangle$ is also a path in G', it follows from the definition of the intervals and Constraint (2) that $[0,1] \subseteq I^+(u'') \cup I^+(u) \cup I^-(v) \cup I^-(v'')$. Moreover, we clearly have $I^+(u'') \subseteq I^+(u')$ and $I^+(v'') \subseteq I^+(v')$, so r must be contained in one of of the intervals $I^+(u'), I^+(u), I^-(v), I^-(v')$, and so p will be cut by C_1.

Now we consider the cut C_2. First consider a path $p = \langle s, u', v, v', t\rangle$ of length 4. Note that $I^+(u') = I^-(u')$ and $I^+(v') = I^-(v')$, and by similar reasoning to the above, we have $[0,1] \subseteq I^+(u') \cup (I^+(v) \cap I^-(v)) \cup I^-(v')$, and so at least one of r_1 or r_2 must be in $I^-(u) \cap I^+(u)$ for some $u \in \{u', v, v'\}$, and this vertex will be cut by C_2.

Finally, let $p = \langle s, u', u, v, v', t\rangle$ be a 5-path in G', and let us see that p is cut by C_2. Since $I^+(u') = I^-(u')$ and $I^+(v') = I^-(v')$, we may assume that $r_1 > x_{u'}$ and $r_2 < 1 - x_{v'}$ (otherwise, u' or v', respectively, will be cut). Thus, $r_1, r_2 \in I^+(u) \cup I^-(v)$. In particular, $r_1 \in I^+(u)$ or $r_2 \in I^-(v)$; otherwise, we would have $r_2 \in I^+(u)$ and r_1 to the left of $I^+(u)$, that is $r_1 < y^+(u) \le x_{u'}$, contradicting our current assumption.

If both radii were in $I^+(u)$ or both in $I^-(v)$, then again the corresponding vertex would be cut by C_2, so assume $r_1 \in I^+(u)$ and $r_2 \in I^-(v)$. We claim that in this case we must have $r_1 \in I^-(u) \cap I^+(u)$ or $r_2 \in I^-(v) \cap I^+(v)$, and then the corresponding vertex will be cut in C_2. For the sake of contradiction, assume $r_1 \in I^+(u) \setminus I^-(u)$ and $r_2 \in I^-(v) \setminus I^+(v)$. The fact that both of these vertices are mapped to two distinct intervals means that there exist vertices u'', v'' in G' such that $\langle s, v'', v\rangle$ and $\langle u, u'', t\rangle$ are paths in G' and $y^-(u) = x_{u''}$ and $y^+(v) = x_{v''}$. As r_1 is to the right of $I^-(u)$ and r_2 is to the left of $I^+(v)$, we have $r_1 > 1 - x_{u''}$ and $r_1 + 1/2 = r_2 < x_{v''}$. Thus, we get $x_{u''} + x_{v''} > 3/2$, contradicting the fact that the LP value of every vertex in G' is at most 3/4. ◀

Next we bound the expected value of the cut.

▶ **Lemma 9.** *Every vertex $v \in V$ is cut by Algorithm 5-Round with probability at most $(4/3) \cdot x_v$.*

Proof. If $x_v \ge 3/4$, this is trivial. Thus, let us look at vertices in $V \setminus C_0$.

First, consider a vertex v such that $x_v < 1/2$. Note that v is cut by C_1 with probability at most $|I^-(v) \cup I^+(v)|$ (it is exactly this probability if both intervals are contained in $[0,1]$, but this is not necessarily the case). Now consider the definition of C_2. As $x_v < 1/2$, Condition 2

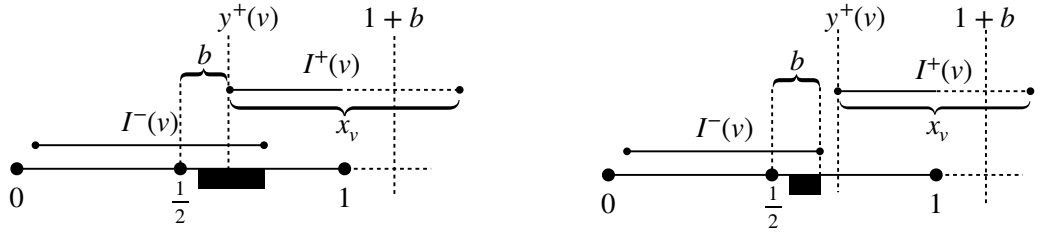


Figure 2 Two examples of the final case in the proof of Lemma 9. The black rectangles denote the range of r_2 for which $v \in C_2$. This range has length at most $|I^-(v) \cap I^+(v)| + b$, while the range of r for which $v \in C_1$ has length at most $|I^-(v) \cup I^+(v)| - b$.

in the algorithm can never occur. Thus the probability that v is cut by C_2 is the probability that it will be cut because of Condition 1, which is exactly $2 \cdot |I^-(v) \cap I^+(v) \cap [0, 1/2]| + 2 \cdot |I^-(v) \cap I^+(v) \cap [1/2, 1]| = 2 \cdot |I^-(v) \cap I^+(v)|$. Thus the overall probability that v is cut is at most

$$\tfrac{2}{3} \cdot |I^-(v) \cup I^+(v)| + \tfrac{1}{3} \cdot 2 \cdot |I^-(v) \cap I^+(v)| = \tfrac{2}{3} \cdot (|I^-(v)| + |I^+(v)|) = \tfrac{4}{3} \cdot x_v.$$

Now consider a vertex $v \in V$ such that $x_v \geq 1/2$ (but less than $3/4$). If Condition 2 never occurs, or only occurs when Condition 1 also occurs (for example, if $I^-(v) = I^+(v)$), then we are done by the same calculation as above. Otherwise, there exist values of r_1 for which Condition 2 occurs but not Condition 1. That is, either

- Case (i): There exists r_1 such that $r_1, r_2 \in I^-(v)$ and r_2 is to the left of $I^+(v)$, or
- Case (ii): There exists r_1 such that $r_1, r_2 \in I^+(v)$ and r_1 is to the right of $I^-(v)$.

Note that at most one of these cases is possible for a given vertex v, since Case (i) implies that $I^-(v)$ intersects the interval $[1/2, 1]$, whereas Case (ii) implies that $I^-(v)$ is strictly to the left of $1/2$. Without loss of generality, assume (only) Case (i) holds. Let $b = \min\{1/2 - y^-(v), y^+(v) - 1/2\}$, that is, the maximum value such that $1/2 + b \in I^-(v)$ and $1/2 + b \leq \min I^+(v)$ (see Figure 2). By our assumption, $b \geq 0$. Thus, $v \in C_2$ *only* if (not iff) $r_2 \in I^-(v) \cap I^+(v)$ or $r_2 \in [1/2, 1/2 + b]$. This happens with probability at most $2|I^-(v) \cap I^+(v)| + 2b$. As we've noted, v is cut by C_1 with probability at most $|I^-(v) \cup I^+(v)|$. However a tighter upper bound on this probability is $|(I^-(v) \cup I^+(v)) \cap [0, 1]|$. Since $x_v \geq 1/2$, $b \leq y^+(v) - 1/2$, and $I^+(v) = [y^+(v), y^+(v) + x_v]$, we have that $|I^+(v) \setminus [0, 1]| \geq b$, and so the total probability that v will be cut is upper bounded by

$$\begin{aligned}
\tfrac{2}{3}|(I^-(v) \cup I^+(v)) \cap [0, 1]| + \tfrac{1}{3} \cdot \big(2|I^-(v) \cap I^+(v)| + 2b\big) \\
&= \tfrac{2}{3}\big(|I^-(v) \cup I^+(v)| - |I^+(v) \setminus [0, 1]| + |I^-(v) \cap I^+(v)| + b\big) \\
&= \tfrac{2}{3}\big(|I^-(v)| + |I^+(v)| - |I^+(v) \setminus [0, 1]| + b\big) \\
&\leq \tfrac{2}{3}\big(|I^-(v)| + |I^+(v)|\big) = \tfrac{4}{3}x_v.
\end{aligned}$$

Figure 2 illustrates this final case. ◀

3 Improved approximations for other small values of L

In this section, we prove Theorem 3. Our approximation is based on an algorithm which takes as a black box a rounding for $(L-1)$-Bounded Cut and converts it into a rounding for L-Bounded Cut. In particular, we show the following result:

▶ **Theorem 10.** *Given a rounding algorithm for $(L-1)$-Bounded Cut which cuts every vertex v with probability at most $\alpha \cdot x_v$, there is a rounding for L-Bounded Cut which cuts every vertex v with probability at most $(1 + \alpha \cdot (1 - 1/(L-2))) \cdot x_v$.*

Theorem 3 now follows easily.

Proof of Theorem 3. Follows immediately by induction, where the base case is Theorem 2 for $L = 6$ (or technically, the more specific Lemma 19), and Theorem 10 gives the inductive step. ◀

We now describe our technique for converting a rounding for $(L-1)$-Bounded Cut to a rounding for L-Bounded Cut. The following algorithm takes as a black box a rounding algorithm A_{L-1} for $(L-1)$-Bounded Cut:

L-Recurse$(G, (x_v)_{v\in V}, A_{L-1})$

- Cut every vertex $v \in V$ such that $x_v \geq 1$, and let V' be the remaining vertices.
- Define $(z_v)_{v\in V'}$ as follows:

$$z_v = \left(1 - \tfrac{1}{L-2}\right) \cdot \tfrac{x_v}{1-x_v}.$$

- Run algorithm A_{L-1} on graph $G|_{V'}$ and LP solution $(z_v)_{v\in V'}$. Let V'' be the remaining vertices (not cut in this step).
- Run **Exact-Round**$(G|_{V''}, (x_v)_{v\in V''})$.

To understand this algorithm, first let us see the feasibility of the solution $(z_v)_{v\in V'}$ defined in the second step.

▶ **Lemma 11.** *If $(x_v)_{v\in V'}$ is a feasible solution for the natural LP relaxation for $(L-1)$-Bounded Cut on a graph with vertex set V', such that $x_v < 1$ for all $v \in V'$, then the solution $(z_v)_{v\in V'}$ defined above is also a feasible LP solution for the same LP relaxation.*

Proof. By our assumption that $x_v < 1$ for all $v \in V'$, we have $z_v \geq 0$ for all vertices. Now let $\langle s, v_1, \ldots, v_{L-2}, t\rangle$ be a path of length $L-1$. By the feasibility of $(x_v)_{v\in V'}$ we know that $\sum_{i=1}^{L-2} x_{v_i} \geq 1$. Therefore, by the convexity of the function $f(x) = x/(1-x)$ for all $x < 1$ (and Jensen's inequality), we have

$$\begin{aligned}\sum_{i=1}^{L-2} z_v &= (L-3) \cdot \tfrac{1}{L-2} \textstyle\sum_{i=1}^{L-2} x_{v_i}/(1-x_{v_i}) \\ &\geq (L-3) \cdot \tfrac{1}{L-2} \textstyle\sum_{i=1}^{L-2} x_{v_i} / \left(1 - \tfrac{1}{L-2} \sum_{i=1}^{L-2} x_{v_i}\right) \\ &\geq (L-3) \cdot \tfrac{1}{L-2} / (1 - \tfrac{1}{L-2}) = 1.\end{aligned}$$

Thus, Constraint (2) is satisfied for this path. For paths of length $L' < L-1$, the argument follows from the above calculation by taking the LP values along the path and appending additional values $x_{L'-1} = \ldots = x_{L-2} = 0$ which are mapped to $z_{L'-1} = \ldots z_{L-2} = 0$. ◀

We can now show our main guarantee for this section.

Proof of Theorem 10. Let us see that Algorithm L-Recurse satisfies the requirements. First note that the algorithm returns a feasible L-bounded cut. Indeed, by Lemma 11, $(z_v)_{v\in V'}$ is a feasible solution for the relaxation for $(L-1)$-Bounded Cut, and so the application of algorithm A_{L-1} will cut all paths of length at most $L-1$ not already cut in the first step. In the remaining graph we have $d(s,t) \geq L$, and so all remaining paths (if any) of length L will be cut in the last step.

As for the approximation guarantee, if $x_v \geq 1$, then the theorem follows trivially. For $v \in V'$, we know that algorithm A_{L-1} will cut v with probability $\alpha'_v \cdot z_v$, for some $\alpha'_v \leq \alpha$. Conditioned on surviving this phase, by Observation 6, v will be cut in the last step with probability at most x_v. Thus the overall probability that v is cut will be at most

$$\begin{aligned}\alpha'_v \cdot z_v + (1 - \alpha'_v \cdot z_v)x_v &= x_v + (1 - x_v)\alpha'_v \cdot z_v \\ &= x_v + \alpha'_v \cdot (1 - 1/(L-2)) \cdot x_v \\ &\leq (1 + \alpha \cdot (1 - 1/(L-2))) \cdot x_v.\end{aligned}$$

◀

4 An $((11/25) \cdot L + O(1))$-approximation

Here we prove Theorem 4, showing that $L(1/2 - o(1))$ is not the best possible approximation for general (bounded) L. Similarly to our algorithm for $L = 5$, we will map every vertex to different intervals corresponding to different positions the vertex can have in an L-bounded path from s to t, and then cut all intervals containing a random radius in $[0, 1]$. Our improvement follows from showing that these intervals will be mapped close together. However, for vertices with small LP value, this will not be sufficient, since the intervals can still be disjoint. See Appendix B for an example. To avoid this problem, we will define a new LP solution which will greatly decrease the LP value of vertices which already have a small LP value, giving us an advantage over previous algorithms for these vertices as well.

Distort-Round

- Cut all vertices $v \in V$ with $x_v \geq 25/(11(L-1))$. Let G' be the graph on remaining vertices.
- For every remaining vertex v, let $i_{\min}(v) = d(s, v)$ and $i_{\max}(v) = L - d(v, t)$ (the first and last possible positions of v in an L-bounded path), where d is the hop-distance in G'. Let $V' = \{v \in G \mid i_{\min}(v) \leq i_{\max}(v)\}$, and on these vertices define $(x'_v)_{v \in V'}$ as follows:

  $$x'_v = \max\{0, (23/20) \cdot x_v - 3/(20(L-1))\}.$$

- For all $v \in V'$ and $i \in \{i_{\min}(v), \ldots, i_{\max}(v)\}$ define the following intervals, where paths are in G' and as before $x'(p)$ is the total x' value of all vertices in p excluding the endpoints:

  $$y_i(v) := \min_{\substack{p: s \rightsquigarrow v \\ |p| \leq i}} x'(p) \qquad \text{and} \qquad I_i(v) = [y_i(v), y_i(v) + x'_v].$$

- Sample $r \in [0, 1]$ uniformly at random.
- Return the cut $\{v \in V' \mid r \in I_i(v) \text{ for some } i\}$.

Let us first see the correctness of the algorithm:

▶ **Lemma 12.** *Algorithm Distort-Round returns a valid L-bounded vertex cut.*

Proof. Since the first step cuts all paths not entirely in G', let us focus on paths in G'. It is straightforward to see that in this graph, the interior vertices of every L-bounded $s - t$ path are all in V', since if v is ith vertex in such a path, then $i_{\min}(v) \leq i \leq i_{\max}(v)$. For any such path we have $\sum_{v \in p \setminus \{s,t\}} x'_v \geq \sum_{v \in p \setminus \{s,t\}} (23/20) \cdot x_v - ((|p| - 1)/(L-1))\, 3/20 \geq 1$.

Let us see that such a path will necessarily be cut. Denote $p = \langle s, v_1, \ldots, v_{L'}, t \rangle$ for some $L' \leq L - 1$. By the definition of our intervals, it can be shown by induction that for every $i \in [L']$ we have $[0, y_i(v_i) + x'_{v_i}] \subseteq \bigcup_{j=1}^{i} I_j(v_j)$. Thus, to see that the path is cut in the last

step, it suffices to show that $y_{L'}(v_{L'}) + x'_{v_{L'}} \geq 1$. But since $(v_{L'}, t) \in E$, by the definition of y, this is the total x' value of some $s-t$ path of length at most $L'+1$, and so it must be at least 1. ◀

To bound the approximation guarantee, we need to bound the probability that any vertex $v \in V'$ is cut. Let $D(v) := i_{\max}(v) + 1 - i_{\min}(v)$ denote the number of intervals defined for v. Then one trivial bound is the following:

▶ **Observation 13.** *Every vertex $v \in V'$ is cut with probability at most $D(v) \cdot x'_v$.*

However, this bound may be too conservative. In fact, if v participates in a large number of intervals, we can show that these intervals cannot be too spread out.

▶ **Lemma 14.** *For any given vertex $v \in V'$, all the intervals $I_i(v)$ are contained in a single interval of length at most $161/110 - D(v) \cdot 271/(110(L-1)) + O(x_v)$.*

Proof. Let p be a shortest path from s to v in G'. By the definition of G', all vertices $u \in p$ have $x_u < 25/(11(L-1))$, and so $x'_u < 271/(110(L-1))$. By definition of $y_i(v)$, this means that for all $i \geq i_{\min}(v)$ we have

$$y_i(v) \leq x'(p) \leq (d(s,v)-1)271/(110(L-1)) = (i_{\min}(v)-1)271/(110(L-1)),$$

and in particular,

$$y_i(v) + x'_v \leq (i_{\min}(v)-1)271/(110(L-1)) + x'_v \tag{3}$$

Similarly, for any $i \in \{i_{\min}(v), \ldots, i_{\max}(v)\}$, let p_i be an $s-v$ path of length $\leq i$ such that $y_i(v) = x'(p_i)$, and let p' be a shortest path from v to t. Then $p_i \circ p'$ is an L-bounded $s-t$ path with x' value

$$\begin{aligned} x'(p_i) + x'_v + x'(p') &\leq y_i(v) + x'_v + (|p'|-1)271/(110(L-1)) \\ &= y_i(v) + x'_v + (L - i_{\max}(v) - 1)271/(110(L-1)) \\ &= y_i(v) + x'_v - i_{\max}(v)271/(110(L-1)) + 271/110. \end{aligned}$$

However, its x' value must also be at least 1, and so we get

$$y_i(v) \geq i_{\max}(v)271/(110(L-1)) - 161/110 - x'_v. \tag{4}$$

Since $I_i(v) = [y_i(v), y_i(v) + x'_v]$, equations (3) and (4) imply that all these intervals are contained in a single interval of length at most

$$\begin{aligned} (i_{\min}(v)-1)271/(110(L-1)) + x'_v - (i_{\max}(v)271/(110(L-1)) - 161/110 - x'_v) \\ = 161/110 - D(v) \cdot 271/(110(L-1)) + O(x_v). \end{aligned}$$ ◀

We can now prove our final guarantee for general (bounded) L:

▶ **Lemma 15.** *Every vertex $v \in V$ is cut by Algorithm Distort-Round with probability at most $((11/25)(L-1) + O(1))x_v$.*

Proof. For vertices $v \notin V'$, this is trivial, so assume $v \in V'$. Also, let us assume $x'_v > 0$, since otherwise v would not be cut. First, from Observation 13 and Lemma 14, we have that the probability that v is cut is at most

$$\min\{D(v) \cdot x'_v,\ 161/110 - D(v) \cdot 271/(110(L-1))\} + O(x_v) \leq \frac{161 x'_v}{110 x'_v + 271/(L-1)} + O(x_v),$$

where the inequality is obtained by maximizing over all possible values of $D(v)$. Let us scale up x_v and denote $c_v = (L-1)x_v$, which (by our assumption that $x'_v \neq 0$) implies that $x'_v = ((23 - 3/c_v)/20)x_v$. Thus, we can rewrite the above upper bound on the probability as

$$\left(\frac{23 - 3/c_v}{20} \cdot \frac{161(L-1)}{11(23c_v - 3)/2 + 271} + O(1)\right) x_v$$

Ignoring the $O(1)$ term, the expression in parentheses is maximized for $c_v = (3+\sqrt{1626/11})/23$, and so is at most $0.43954(L-1) < (11/25) \cdot (L-1)$. ◀

5 A 7/4-approximation for $L = 6$

We now prove Theorem 2. We note first that the techniques we have seen already give an improvement over the previous 3-approximation for Minimum 6-Bounded Node Cut. Indeed, Lemma 9 and Theorem 10 together give a 2-approximation.

As a warm-up to our final algorithm, let us first see an alternative 2-approximation which does not use Theorem 10 or our algorithm for $L = 5$. We introduce some notation which we will use both for our warm-up algorithm for $L = 6$ as well as our main algorithm.

Similarly to our algorithm for $L = 5$, we will map every vertex to a small number of possible intervals, corresponding to its possible positions in a 6-bounded path, relative to s and t. For all $i \in [3]$, and all vertices $u, v \in V \setminus \{s, t\}$ such that $d_G(s, u) \leq i$ and $d_G(v, t) \leq i$, define

$$y_i^G(u) := \min_{\substack{p: s \rightsquigarrow u \\ |p| \leq i}} x(p) \qquad \text{and} \qquad y_{-i}^G(v) := 1 - \min_{\substack{p: v \rightsquigarrow t \\ |p| \leq i}} x(p).$$

As before, if $(s, u), (v, t) \in E$, for such neighbors of s, t we define

$$I_1^G(u) = [0, x_u] \qquad I_{-1}^G(v) = [1 - x_v, 1].$$

For other vertices v, wherever the relevant $y_i^G(v)$ values are defined, we also define intervals

$$I_2^G(v) = \left[y_2^G(v), y_2^G(v) + x_v\right] \qquad I_3^G(v) = \left[y_3^G(v), y_{-3}^G(v)\right] \qquad I_{-2}(v) = \left[y_{-2}^G(v) - x_v, y_{-2}^G(v)\right].$$

We will drop the superscript G from the above definitions when it is clear from the context. If $I_1(v)$ (resp. $I_{-1}(v)$) is defined, we do not define any other interval for v, as such an interval would be contained in $I_1(v)$ (resp. $I_{-1}(v)$). Note that every interval associated with a vertex v has length at most x_v (in fact, exactly x_v except for $I_3(v)$). As before, it is not hard to see that the left (resp. right) endpoints of the intervals $I_2(v), I_3(v), I_{-2}(v)$ (or whichever subsequence of intervals is defined for this vertex) form a monotone non-increasing sequence.

Consider the following rounding algorithm.

Algorithm Simple-6-Round

- Cut all vertices v with $x_v \geq 1/2$, and let $G^{1/2}$ be the remaining graph after deleting these vertices.
- Sample $r \in [0, 1]$ uniformly at random.
- Cut all vertices v such that $r \in I_i^{G^{1/2}}(v)$ for some $i \in \{1, 2, 3, -2, -1\}$.

One can check easily that this algorithm will cut all 6-bounded $s-t$ paths (the proof is nearly identical to the first part of the proof of Lemma 8). To see that it gives a 2-approximation, first note that trivially every vertex v with $x_v \geq 1/2$ or for which at most two of the intervals $I_2^{G^{1/2}}(v), I_3^{G^{1/2}}(v), I_{-2}^{G^{1/2}}(v)$ are defined, will be cut with probability at most $2x_v$. Thus we only need to concern ourselves with vertices for which all three intervals are defined. Since we removed vertices with LP value at least $1/2$, this means that in the remaining graph we have $y_2^{G^{1/2}}(v) < 1/2$ and $y_{-2}^{G^{1/2}}(v) > 1/2$, and so by monotonicity, $I_2^{G^{1/2}}(v)$ and $I_{-2}^{G^{1/2}}(v)$ must intersect, and $I_3^{G^{1/2}}(v)$ must be contained in their union. Thus, the union of the three intervals has length at most $2x_v$, which bounds the probability of cutting v.

To improve over this algorithm, we must make a number of changes. First of all, we cannot cut all vertices with $x_v \geq 1/2$. We must allow some (at least slightly) costlier vertices to remain. This means that we will not have the nice overlap property described above. We can overcome this by avoiding a small subinterval in the middle of $[0,1]$ in our radius sampling. However, this does not resolve the issue that some vertices will still be mapped to two possibly disjoint intervals, and in fact will exacerbate the problem by increasing the probability of hitting any given interval (since we are restricting our radius to a smaller sample space). We are able to overcome all these pitfalls by choosing at random either a single radius or a two-radius cut as we did in Algorithm 5-Round, and defining our two-radius cut carefully.

Algorithm 6-Round

- Let $C_0 = \{v \in V \mid x_v \geq 4/7\}$. Let G' be the remaining graph after deleting all vertices in C_0.
- Sample $r \in [0, 3/7] \cup [4/7, 1]$ uniformly at random.
- Sample $r_1 \in [0, 3/7]$ uniformly at random, and let $r_2 = r_1 + 4/7$.
- Let C_1 be the set of all vertices v such that $r \in I_i^{G'}(v)$ for some $i \in \{1, 2, 3, -2, -1\}$.
- Let C_2 be the set of all vertices v satisfying at least one of the following conditions:
  1. $r_1 \in I_1^{G'}(v)$ or $r_2 \in I_{-1}^{G'}(v)$.
  2. $r_1 \in I_2^{G'}(v)$.
  3. $r_1 \in I_3^{G'}(v)$ and $\frac{3}{7} \in I_3^{G'}(v)$.
  4. $r_2 \in I_{-2}^{G'}(v)$ and $\frac{3}{7} \in I_{-2}^{G'}(v)$.
- Let C_3 be the set of all vertices v satisfying at least one of the following conditions:
  1. $r_1 \in I_1^{G'}(v)$ or $r_2 \in I_{-1}^{G'}(v)$.
  2. $r_2 \in I_{-2}^{G'}(v)$.
  3. $r_2 \in I_3^{G'}(v)$ and $\frac{4}{7} \in I_3^{G'}(v)$.
  4. $r_1 \in I_2^{G'}(v)$ and $\frac{4}{7} \in I_2^{G'}(v)$.
- Return a random cut according to the following distribution: $C_0 \cup C_1$ with probability $1/2$, $C_0 \cup C_2$ with probability $1/4$, and $C_0 \cup C_3$ with probability $1/4$.

First let us see why this is a valid 5-bounded cut.

▶ **Lemma 16.** *Algorithm 6-Round cuts all paths of length at most* 6 *from s to t.*

Proof. As we did for $L = 5$, we focus on paths in G'. Also, the proof of the correctness of $C_0 \cup C_1$ is essentially the same as the proof of correctness of the corresponding cut in Lemma 8.

Since C_2 and C_3 are symmetrically defined, let us focus on C_2. Let $\langle s, u, u', w, v', v, t\rangle$ be a 5-path from s to t in G'. The proof for paths of length < 5 is similar or simpler. As before, it can easily be seen that $[0,1] \subseteq I_1(u) \cup I_2(u') \cup I_3(w) \cup I_{-2}(v') \cup I_{-1}(v)$. If the first condition

in the definition of C_2 does not hold w.r.t. $I_1(u)$ or $I_{-1}(v)$, then r_1 and r_2 must both intersect the intervals $I_2(u') \cup I_3(w) \cup I_{-2}(v')$. If in addition, the second condition does not hold w.r.t. u, then both these radii (in fact all points in $[r_1, r_2]$) intersect the intervals $I_3(w) \cup I_{-2}(v')$. Since $r_2 - r_1 = 4/7$ and all intervals corresponding to vertices in G' have length less than $4/7$, this necessarily means that $r_1 \in I_3(w)$ and $r_2 \in I_{-2}(v')$. Since $\frac{3}{7} \in [r_1, r_2]$, either $\frac{3}{7} \in I_3(w)$ or $\frac{3}{7} \in I_{-2}(v')$. In the first case, $w \in C_2$ by Condition 3, and in the second, $\frac{3}{7} \in I_{-2}(v')$ and so $v' \in C_2$ by Condition 4. ◀

Before bounding the expected value of the cut, we note that, as before (in Algorithm **Simple-6-Round**), Algorithm **6-Round** also has the property that for every $v \notin C_0$, at most two intervals are responsible for v being in the single radius cut.

▶ **Lemma 17.** *For $v \notin C_0$, If $I_{-2}^{G'}(v)$ and $I_2^{G'}(v)$ are both defined, then the random radius r is in C_1 iff it is in $I_{-2}^{G'}(v) \cup I_2^{G'}(v)$.*

Proof. Note that in this case, $I_3^{G'}(v)$ is also defined. Since all three intervals are defined for v (and $I_1^{G'}(v), I_{-1}^{G'}(v)$ are not defined – so $d(s,v), d(v,t) > 1$), there must exist paths $\langle s, u, v\rangle$ and $\langle v, v', t\rangle$ in G', and by the bound on LP values of vertices in G', we have $y_2(v) \le x_u < 4/7$ and $y_{-2}(v) \ge 1 - x_{v'} > 3/7$. Thus, by the monotonicity of the interval sequence, $I_3(v) \setminus (3/7, 4/7)$ is contained in $(I_{-2}(v) \cup I_2(v)) \setminus (3/7, 4/7)$, and the lemma follows. ◀

Since this is the only case in which v will be mapped to more than two intervals, this immediately bounds the probability that such a vertex participates in C_1:

▶ **Corollary 18.** *For every $v \notin C_0$, the probability that $v \in C_1$ is at most $\frac{7}{6} \cdot 2x_v$.*

Now let us bound the expected value of the cut in our final algorithm.

▶ **Lemma 19.** *Every vertex $v \in V$ is cut by Algorithm 6-**Round** with probability at most $(7/4) \cdot x_v$.*

Proof. If $x_v \ge 4/7$, this is trivial. Thus, let us look at vertices cut by C_1, C_2, or C_3.

For convenience, define a random set

$$C' = \begin{cases} C_2, & \text{with probability } 1/2 \\ C_3, & \text{otherwise.} \end{cases}$$

Thus the final step in the algorithm can be equivalently stated as returning $C_0 \cup C_1$ w.p. $1/2$ and $C_0 \cup C'$ w.p. $1/2$.

To analyze the probability that a vertex v in G' will be cut overall, we will consider a number of different of cases for this vertex.

Case 1: v is mapped to only one interval. Suppose $I_1(v)$ or $I_{-1}(v)$ are defined. Then $v \in C'$ with probability at most $\frac{7}{3} \cdot x_v$ (by Condition 1 in both C_2 and C_3). In total, it will be cut with probability at most $\left(\frac{1}{2} \cdot \frac{7}{6} + \frac{1}{2} \cdot \frac{7}{3}\right) x_v = \frac{7}{4} \cdot x_v$. This holds similarly for any vertex v which is not necessarily a neighbor of s or t but which is mapped to only one interval.

Case 2: v is mapped to at least two intervals, and both $I_{-2}(v)$ and $I_2(v)$ are defined for v. Again, note that this implies that $I_3(v)$ is also defined and $I_1(v), I_{-1}(v)$ are not defined. Recall that $I_3(v) = [y_3(v), y_{-3}(v)]$ and as noted above, in this case we must have $y_{-3}(v) \ge y_{-2}(v) \ge \frac{3}{7}$ and $y_3(v) \le y_2(v) \le \frac{4}{7}$. Thus, $I_3(v)$ cannot be entirely to the left of $3/7$ or entirely to the right of $4/7$. This together with monotonicity of intervals gives us

the following implications: Conditions 2 and 3 of C_2 each imply $r_1 \in I_3(v) \cap \left[0, \frac{3}{7}\right)$ and Conditions 2 and 3 of C_3 each imply $r_2 \in I_3(v) \cap \left[\frac{4}{7}, 1\right)$, giving two possible ways v may be in C':

- $r_1 \in I_3(v) \cap \left[0, \frac{3}{7}\right)$ and $C' = C_2$, or
- $r_2 \in I_3(v) \cap \left[\frac{4}{7}, 1\right)$ and $C' = C_3$.

Note that the probability that at least one of these events occurs is at most $\frac{7}{6} \cdot |I_3(v)|$.

The final reason we may have $v \in C'$ is due to Condition 4 in either C_2 or C_3. That is, if one of the following occurs:

(a) $r_1 \in I_2(v) \cap \left[0, \frac{3}{7}\right)$ and $C' = C_3$, or

(b) $r_2 \in I_{-2}(v) \cap \left[\frac{4}{7}, 1\right)$ and $C' = C_2$.

Note that each of these two events implies that the given radius (r_1 in (a) and r_2 in (b)) is in fact in $I_2(v) \cap I_{-2}(v)$. Indeed, take event (b) for example. As we've noted, $y_2(v)$ (the left endpoint of $I_2(v)$) is at most $\frac{4}{7}$, so r_2 cannot be to the left of $I_2(v)$. On the other hand, by monotonicity of intervals, since $r \in I_{-2}(v)$, it also cannot be to the right of $I_2(v)$. Thus, $r_2 \in I_2(v) \cap I_{-2}(v)$. Thus, the probability that (a) or (b) occurs is at most $\frac{7}{6} \cdot |I_{-2}(v) \cap I_2(v)|$.

Combining these bounds with Lemma 17 for C_1, we can bound the total probability that v will be cut by

$$\frac{1}{2} \cdot \frac{7}{6} \cdot |I_2(v) \cup I_{-2}(v)| + \frac{1}{2} \cdot \left(\frac{7}{6} \cdot |I_3(v)| + \frac{7}{6} \cdot |I_2(v) \cap I_{-2}(v)| \right)$$
$$= \frac{1}{2} \cdot \frac{7}{6} \cdot (|I_3(v)| + |I_2(v)| + |I_{-2}(v)|) \leq \frac{7}{4} \cdot x_v.$$

Case 3: v is mapped to at two distinct intervals, $I_3(v)$ is defined, but only one of $I_{-2}(v)$, $I_2(v)$ is defined. As before, because v is mapped to two distinct intervals, Condition 1 for both C_2 and C_3 is irrelevant. Without loss of generality, assume only $I_2(v)$ and $I_3(v)$ are defined for v, and note that Condition 4 for C_2 and Condition 2 for C_3 are now also irrelevant. Thus, v can be in C' only if Conditions 2 or 3 for C_2 or Conditions 3 or 4 for C_3 occur. We further divide this case (under this assumption) into several subcases.

Subcase 3a: $\frac{4}{7} \notin I_2(v)$. As we've noted, $I_2(v)$ cannot lie entirely to the right of $\frac{4}{7}$, so in this subcase it must lie entirely to the left of $\frac{4}{7}$, and by monotonicity, so must $I_3(v)$. Thus, in fact v can only be in C' when $C' = C_2$. For C_2 consider two possibilities. If $\frac{3}{7} \notin I_3(v)$, then only Condition 2 can apply. Otherwise, we have $\frac{3}{7} \in I_3(v)$ and so by monotonicity Condition 2 would imply Condition 3, and so we need only consider Condition 3. Either way, only one interval is responsible for v being in C', and the probability that this occurs is at most $\frac{7}{6} \cdot x_v$. This together with Corollary 18 gives the desired bound on the probability that v is cut.

Subcase 3b: $\frac{4}{7} \in I_2(v)$ and $\frac{3}{7} \in I_3(v)$. In this subcase, we need to be slightly more precise for C_1, and note that the probability that $v \in C_1$ is at most $\frac{7}{6}|I_2(v) \cup I_3(v)|$. As above, because $\frac{3}{7} \in I_3(v)$, Condition 2 for C_2 implies Condition 3 for C_2, so the relevant conditions are Condition 3 in both C_2 and C_3, and Condition 4 in C_3. The probability that $v \in C'$ because of Condition 3 in either set is clearly at most $\frac{7}{6} \cdot x_v$. Similarly, Condition 4 for C_3 also implies that $r_1 \in I_3(v)$, so the probability that $C' = C_3$ and Condition 4 occurs is at most $\frac{7}{6} \cdot |I_2(v) \cap I_3(v)|$. Putting these three bounds together, the probability that v is cut is at most

$$\tfrac{1}{2} \cdot \tfrac{7}{6} \cdot |I_2(v) \cup I_3(v)| + \tfrac{1}{2} \cdot \left(\tfrac{7}{6} \cdot x_v + \tfrac{7}{6} \cdot |I_2(v) \cap I_3(v)| \right) = \tfrac{1}{2} \cdot \tfrac{7}{6} \cdot (x_v + |I_2(v)| + |I_3(v)|) \leq \tfrac{7}{4} \cdot x_v.$$

Subcase 3c: $\frac{4}{7} \in I_2(v)$ and $I_3(v)$ lies to the right of $\frac{3}{7}$. In this subcase, only Condition 3 for C_3 is relevant, and so $v \in C'$ with probability at most $\frac{7}{6} \cdot x_v$, which as we've noted (see Case 3a) is enough.

Subcase 3d: $\frac{4}{7} \in I_2(v)$ and $I_3(v)$ lies to the left of $\frac{3}{7}$. Note that in this case we will have $v \in C'$ precisely when $r_1 \in I_2(v) \cap [0, \frac{3}{7}]$ (regardless of whether $C' = C_2$ or $C' = C_3$), which occurs with probability $\frac{7}{3} \cdot |I_2(v) \cap [0, \frac{3}{7}]|$. Thus, we may also assume $\frac{3}{7} \in I_2(v)$, since otherwise we will never have $v \in C'$, and the probability that v is cut will be much better than we require. However, this means that $[\frac{3}{7}, \frac{4}{7}] \subseteq I_2(v)$, so we can improve our probability bounds by not charging for this omitted interval. That is, the probability that $v \in C_1$ is at most $\frac{7}{6} \cdot \left(|(I_2(v) \cup I_3(v)) \cap [0,1]| - \frac{1}{7}\right)$, and the probability that $v \in C'$ is $\frac{7}{3} \cdot (|I_2(v) \cap [0, \frac{4}{7}]| - \frac{1}{7})$.

Now, suppose first that $x_v \leq \frac{3}{7}$. Then by the above bounds, the probability that v is cut is at most

$$\begin{aligned} \frac{1}{2} \cdot \frac{7}{6} \cdot \left(|I_2(v)| + |I_3(v)| - \frac{1}{7}\right) + \frac{1}{2} \cdot \frac{7}{3} \cdot \left(|I_2(v)| - \frac{1}{7}\right) &\leq \frac{7}{3} \cdot x_v - \frac{1}{4} \\ &\leq \frac{7}{4} \cdot x_v. \qquad \text{since } x_v \leq \frac{3}{7} \end{aligned}$$

Otherwise, we have $x_v \geq \frac{3}{7}$ (and, since $v \notin C_0$, $x_v < \frac{4}{7}$), and so the proof will follow if we can show that v is cut with probability at most $\frac{3}{4}$. To see that this holds, recall that $I_2(v) = [y_2(v), y_2(v) + x_v]$. There are three ways that v might be cut:

- In C_1:
  - If $r \in [0, \frac{3}{7}]$ (where possibly $r \in I_2(v) \cup I_3(v)$) – with probability $\frac{1}{2}$.
  - If $r \in [\frac{4}{7}, y_2(v) + x_v]$ (where $r \in I_2(v)$) – with probability $\frac{7}{6} \cdot \left(y_2(v) + x_v - \frac{4}{7}\right) < \frac{7}{6} \cdot y_2(v)$.
- In C':
  - If $r_1 \in [y_2(v), \frac{3}{7}]$ (where $r_1 \in I_2(v)$) – with probability $\frac{7}{3} \cdot (\frac{3}{7} - y_2(v)) = 1 - \frac{7}{3} \cdot y_2(v)$.

And indeed, as required, the probability that at least one of these events happens is at most

$$\frac{1}{2} \cdot \left(\frac{1}{2} + \frac{7}{6} \cdot y_2(v)\right) + \frac{1}{2} \cdot \left(1 - \frac{7}{3} \cdot y_2(v)\right) = \frac{3}{4} - \frac{7}{6} \cdot y_2(v) \leq \frac{3}{4}. \qquad \blacktriangleleft$$

References

1. J. Adámek and V. Koubek. Remarks on flows in network with short paths. *Commentationes Mathematicae Universitatis Carolinae*, 12(4):661–667, 1971.
2. K. Altmanová, P. Kolman, and J. Voborník. On polynomial-time combinatorial algorithms for maximum l-bounded flow. In *Proc. of Algorithms and Data Structures Symposium (WADS)*, pages 14–27, 2019.
3. G. Baier, T. Erlebach, A. Hall, E. Köhler, P. Kolman, O. Pangrác, H. Schilling, and M. Skutella. Length-bounded cuts and flows. *ACM Trans. Algorithms*, 7(1):4:1–4:27, 2010. Preliminary version in *Proc. of ICALP*, 2006.
4. M. O. Ball, B. L. Golden, and R. V. Vohra. Finding the most vital arcs in a network. *Operations Research Letters*, 8(2):73–76, 1989.
5. A. Bar-Noy, S. Khuller, and B. Schieber. The complexity of finding most vital arcs and nodes. Technical Report CS-TR-3539, Univ. of Maryland, Dept. of Computer Science, November 1995. URL: `ftp://ftp.cs.umd.edu/pub/papers/papers/3539/3539.ps.Z`.
6. C. Bazgan, A. Nichterlein, and R. Niedermeier. A refined complexity analysis of finding the most vital edges for undirected shortest paths. In *Proc. of Algorithms and Complexity - 9th International Conference (CIAC)*, volume 9079 of *LNCS*, pages 47–60, 2015.

7 J. A. Bondy and U. S. R. Murty. *Graph Theory with Applications*. North Holland, New York, 1976.

8 I. Dinur and S. Safra. On the hardness of approximating minimum vertex cover. *Annals of Mathematics*, 162(1):439–485, 2005. doi:10.4007/annals.2005.162.439.

9 P. Dvořák and D. Knop. Parametrized complexity of length-bounded cuts and multi-cuts. In *Proc. of 12 Annual Conference on Theory and Applications of Models of Computation (TAMC)*, pages 441–452, 2015.

10 T. Fluschnik, D. Hermelin, A. Nichterlein, and R. Niedermeier. Fractals for kernelization lower bounds. *SIAM J. Discrete Math*, 32(1):656–681, 2018.

11 P. A. Golovach and D. M. Thilikos. Paths of bounded length and their cuts: Parameterized complexity and algorithms. *Discrete Optimization*, 8(1):72–86, 2011.

12 S. Khot, D. Minzer, and M. Safra. Pseudorandom sets in grassmann graph have near-perfect expansion. In *59th IEEE Annual Symposium on Foundations of Computer Science (FOCS)*, pages 592–601, 2018. doi:10.1109/FOCS.2018.00062.

13 S. Khot and O. Regev. Vertex cover might be hard to approximate to within 2. *Journal of Computer and System Sciences*, 74(3):335–349, 2008. Computational Complexity 2003. doi:10.1016/j.jcss.2007.06.019.

14 P. Kolman. On algorithms employing treewidth for l-bounded cut problems. *Journal of Graph Algorithms and Applications*, 22(2):177–191, 2018. doi:10.7155/jgaa.00462.

15 E. Lee. Improved hardness for cut, interdiction, and firefighter problems. In *Proc. of 44rd International Colloquium on Automata, Languages, and Programming (ICALP)*, 2017.

16 L. Lovász, V. Neumann-Lara, and M. D. Plummer. Mengerian theorems for paths of bounded length. *Periodica Mathematica Hungarica*, 9:269–276, 1978.

17 A. R. Mahjoub and S. T. McCormick. Max flow and min cut with bounded-length paths: complexity, algorithms, and approximation. *Math. Program*, 124(1-2):271–284, 2010.

18 F. Pan and A. Schild. Interdiction problems on planar graphs. *Discrete Applied Mathematics*, 198:215–231, 2016.

A Hardness of L-Bounded Cut for $L \geq 5$

The 1.1377-hardness of approximation result of Baier et al. [3] is based on a reduction from Vertex Cover. Given an instance $G = (V, E)$ of Vertex Cover, they construct an instance (G', s, t) of L-Bounded Cut such that, given a vertex cover of size x in G, one can efficiently construct an L-bounded cut of size $|V| + x$ in G', and vice versa. This yields the following general hardness:

▶ **Theorem 20** (Implicit in [3]). *For any $0 \leq c \leq s \leq 1$, approximating L-Bounded Edge Cut for $L \geq 4$ or L-Bounded Node Cut for $L \geq 5$ to within $(1+s)/(1+c)$ is at least as hard as Gap Vertex Cover(c, s).*

The 1.1377-hardness of approximation of L-Bounded Cut is in fact the hardness one gets from the above theorem by plugging in the NP-hardness of Gap Vertex Cover for $c = (\sqrt{5}-1)/2 + \varepsilon$ and $s = (71 - 31\sqrt{5})/2 - \varepsilon$, implied by the work of Dinur and Safra [8].

However, we can also plug in newer or different hardness results for Vertex Cover in the above theorem. For instance, the NP-hardness of Gap Vertex Cover has since been improved to include the case of $c = 1/\sqrt{2} + \varepsilon$, $s = 1 - \varepsilon$ [12], which improves the hardness of approximation of L-Bounded Cut to $2/(1 + 1/\sqrt{2}) - \varepsilon < 1.1715$. Also, plugging in the Unique Games hardness of Gap Vertex Cover$(1/2 + \varepsilon, 1 - \varepsilon)$ [13] gives a $(4/3 - \varepsilon)$ Unique Games hardness of approximation for L-Bounded Cut, which is now matched by our algorithm for 4-Bounded Edge Cut and 5-Bounded Node Cut.

B Short intervals may be disjoint

We motivate the distorted LP values x'_v in algorithm **Distort-Round** by giving an example in which, if the original LP values x_v are used instead of x'_v, some vertices may be cut with probability at least $L/2 - O(1)$ times their LP value.

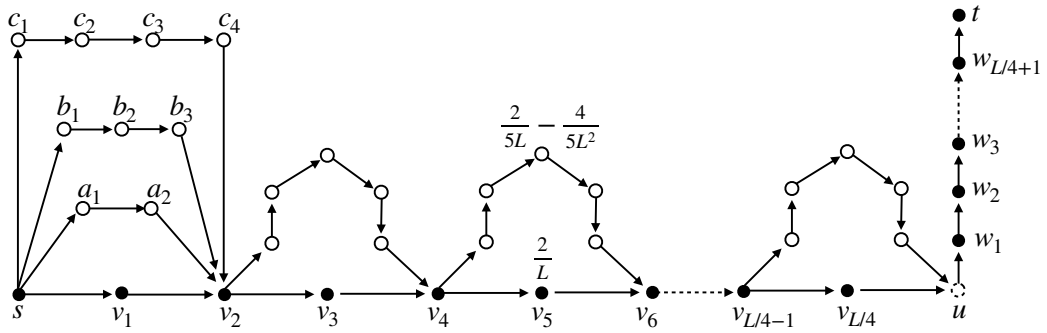


Figure 3 Motivating example for Algorithm Distort-Round.

Consider the graph in Figure 3, and suppose we are given the following LP values: $x_{v_i} = 2/L$ for all $i \in [L/4]$, $x_{w_i} = 2/L$ for all $i \in [L/4+1]$, $x_{a_1} = x_{a_2} = 1/L - 1/(2L^2)$, $x_{b_1} = x_{b_2} = x_{b_3} = 2/(3L) - 2/(3L^2)$, $x_{c_i} = 1/(2L) - 3/(4L^2)$ for all $i \in [4]$, all other white vertices have LP value $2/(5L) - 4/(5L^2)$, and finally $x_u = 1/(2L^2)$.

The following is easy to check:

1. These values are a feasible LP solution.
2. The vertex u may have all possible positions in an L-bounded path from $i_{\min}(u) = L/4+1$ up to $i_{\max}(u) = 3L/4 - 2$.
3. Defining $y_i(u)$ analogously to algorithm **Distort-Round** with values x_v instead of x'_v, for all $k \in [L/2 - 3]$ we have $y_{L/4+k} = 1/2 - (k-1)/L^2$, and $I_{L/4+k} = [y_{L/4+k}, y_{L/4+k} + 1/(2L^2)]$.

Thus, u is indeed mapped to $L/2 - 3$ disjoint intervals, and the probability that it would be cut by a uniformly chosen radius as in algorithm **Distort-Round** is $(L/2 - 3)x_u$.

On the Facility Location Problem in Online and Dynamic Models

Xiangyu Guo
Department of Computer Science and Engineering, University at Buffalo, NY, USA
xiangyug@buffalo.edu

Janardhan Kulkarni
The Algorithms Group, Microsoft Research, Redmond, WA, USA
jakul@microsoft.com

Shi Li
Department of Computer Science and Engineering, University at Buffalo, NY, USA
shil@buffalo.edu

Jiayi Xian
Department of Computer Science and Engineering, University at Buffalo, NY, USA
jxian@buffalo.edu

Abstract

In this paper we study the facility location problem in the online with recourse and dynamic algorithm models. In the online with recourse model, clients arrive one by one and our algorithm needs to maintain good solutions at all time steps with only a few changes to the previously made decisions (called recourse). We show that the classic local search technique can lead to a $(1+\sqrt{2}+\epsilon)$-competitive online algorithm for facility location with only $O\left(\frac{\log n}{\epsilon}\log\frac{1}{\epsilon}\right)$ amortized facility and client recourse, where n is the total number of clients arrived during the process.

We then turn to the dynamic algorithm model for the problem, where the main goal is to design fast algorithms that maintain good solutions at all time steps. We show that the result for online facility location, combined with the randomized local search technique of Charikar and Guha [8], leads to a $(1+\sqrt{2}+\epsilon)$-approximation dynamic algorithm with total update time of $\tilde{O}(n^2)$ in the incremental setting against adaptive adversaries. The approximation factor of our algorithm matches the best offline analysis of the classic local search algorithm.

Finally, we study the fully dynamic model for facility location, where clients can both arrive and depart. Our main result is an $O(1)$-approximation algorithm in this model with $O(|F|)$ preprocessing time and $O(n\log^3 D)$ total update time for the HST metric spaces, where $|F|$ is the number of potential facility locations. Using the seminal results of Bartal [3] and Fakcharoenphol, Rao and Talwar [12], which show that any arbitrary N-point metric space can be embedded into a distribution over HSTs such that the expected distortion is at most $O(\log N)$, we obtain an $O(\log|F|)$ approximation with preprocessing time of $O(|F|^2\log|F|)$ and $O(n\log^3 D)$ total update time. The approximation guarantee holds in expectation for every time step of the algorithm, and the result holds in the oblivious adversary model.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Theory of computation → Online algorithms

Keywords and phrases Facility location, online algorithm, recourse

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.42

Category APPROX

Related Version Full version at `https://arxiv.org/abs/2002.10658`

Funding X. Guo, S. Li and J. Xian are partially supported by NSF grants CCF- 1717134 and CCF-1844890.



© Xiangyu Guo, Janardhan Kulkarni, Shi Li, and Jiayi Xian;
licensed under Creative Commons License CC-BY



1 Introduction

In the *(uncapacitated) facility location problem*, we are given a metric space $(F \cup C, d)$, where F is the set of facility locations, C is the set of clients, and $d : (F \cup C) \times (F \cup C) \to \mathbb{R}_{\geq 0}$ is a distance function, which is non-negative, symmetric and satisfies triangle inequalities. For each location $i \in F$, there is a facility opening cost $f_i \geq 0$. The goal is open a subset $S \subseteq F$ of facilities so as to minimize cost of opening the facilities and the connection cost. The cost of connecting a client j to an open facility i is equal to $d(j, i)$. Hence, the objective function can be expressed concisely as $\min_{S \subseteq F} \left(f(S) + \sum_{j \in C} d(j, S) \right)$, where for a set $S \subseteq F$, $f(S) := \sum_{i \in S} f_i$ is the total facility cost of S and $d(j, S) := \min_{i \in S} d(j, i)$ denotes the distance of j to the nearest location in S. The facility location problem arises in countless applications: in the placement of servers in data centers, network design, wireless networking, data clustering, location analysis for placement of fire stations, medical centers, and so on. Hence, the problem has been studied extensively in many different communities: approximation algorithms, operations research, and computational geometry. In the approximation algorithms literature in particular, the problem occupies a prominent position as the development of every major technique in the field is tied to its application on the facility location problem. See the text book by Williamson and Shmoys [27] for more details. The problem is hard to approximate to a factor better than 1.463 [20]. The current best-known polynomial-time algorithm is given by the third author, and achieves 1.488-approximation [23].

Motivated by its applications in network design and data clustering, Meyerson [24] initiated the study of facility location problem in the online setting. Here, clients arrive online one-by-one, the algorithm has to assign the newly arriving client to an already opened facility or needs to open a new facility to serve the request. The decisions made by the algorithm are *irrevocable*, in the sense that a facility that is opened cannot be closed and the clients cannot be reassigned. In the online setting, Meyerson [24] designed a very elegant randomized algorithm that achieves an $O(\log n)$ competitive ratio, and also showed that no online algorithm can obtain $O(1)$ competitive ratio. This result was later extended by Fotakis [15] to obtain an *asymptotically optimal* $O(\log n / \log \log n)$-competitive algorithm. Both the algorithms and analysis techniques in [15, 24] were influential, and found many applications in other models such as streaming [17]. The lowerbound in Fotakis [15] holds even in very special metric spaces such as HSTs or the real line. Since then, several online algorithms have been designed achieving the same competitive ratio with more desirable properties such as deterministic [1], primal-dual [14], or having a small memory footprint [16]. We refer to a beautifully written survey by Fotakis [17] for more details.

The main reason to assume that decisions made by an algorithm are irrevocable is because the cost of changing the solution is expensive in some applications. However, if one examines these above applications closely, say for example connecting clients to servers in data centers, it is more natural to assume that decisions need not be irrevocable but the algorithm *should not change the solution too much*. This is even more true in modern data centers where topologies can be reconfigured; see [18] for more details. There are two standard ways of enforcing that an algorithm does not make too many changes to the solution:

- **Recourse.** The recourse per step of an online algorithm is the *number of changes* it makes to the solution. Recourse captures the *minimal* amount of changes an online algorithm *has to make* to maintain a desired competitive ratio due to the *information theoretic* limits. For the facility location problem, depending on the application, the recourse can correspond to: 1) the number of changes made to the opened facilities (called *facility*

recourse) 2) the number of reconnections made to the clients (called *client recourse*). Notice that we can assume for every facility we open/close, we have to connect/disconnect at least one client. Thus the client recourse is at least the facility recourse.

- **Update time.** While having a small recourse is enough in some applications, it is not enough in some others. Take wireless networks as a concrete example. Here, the set of clients (mobile devices) keeps changing over time, and it is necessary to *update* the assignment of clients to facilities as *quickly* as possible so to minimize the service disruption. This motivates the *dynamic facility location problem*, which is similar to the one in online setting except that at each time step either *a new client arrives or an existing client departs*. The goal is to always maintain a solution that is a constant factor approximation to the optimal solution, while minimizing *the total time spent in updating the solution.* We emphasize that we require our dynamic algorithms to maintain *an actual assignment of clients to facilities*, not just the set of open facilities and an estimate of connection cost. This is important for applications mentioned above.

Recently, these applications have motivated the study of the facility location problem, and its closely related k-clustering problems, in the framework of recourse and dynamic algorithms [13, 10, 19, 9, 22, 21, 7]. The models for facility location problem considered in [13, 10] are different from what we study here, so we discuss them in Section 1.3; we also defer the discussion of k-clustering results [22, 21, 7] to Section 1.3. Goranci *et al* [19] showed that for metric spaces with *doubling dimension* κ, there is a deterministic *fully* dynamic algorithm with $\tilde{O}(2^{\kappa^2})$ update time, which maintains a constant approximation. Unfortunately, this result does not imply any non-trivial bound for the general metric spaces. More recently, Cohen-Addad *et al* [9] studied the facility location problem in general metric spaces but when the cost of opening facilities are same; that is, when $f_i = f, \forall i \in F$. They showed that there exists a *fully* dynamic algorithm that computes a $O(1)$ approximation with update time of $O(n \log n)$ and facility recourse of $O(1)$ per step. They also showed matching lowerbounds for both these results. However, their positive results do not imply any non-trivial bounds to the case when the cost of opening facilities are *non-uniform*. This is an important consideration in network design problems, and is also the form studied in Goranci *et al* [19] and in classic offline literature [27]. Further, to our understanding, none of the existing results bound the *client recourse*, which is an important consideration in real-world problems. As a concrete example, consider the problem of connecting clients to servers in datacenters, which was the original motivation for Meyerson [24] to initiate the study of online facility location problem. Here, it is important that one does not reconnect clients to servers too many times, as such changes can incur significant costs both in terms of disruption of service and the labor cost. Hence a natural question that arises is:

Are there online and dynamic algorithms for facility location problem in arbitrary metric spaces when the facility opening costs are not uniform, which achieve small client recourse and update times?

1.1 Our Contributions

In the following theorems, we use n to denote the total number of facility locations and all clients that ever arrived, $|F|$ the number of facilities, and D to denote the diameter of the metric d (assuming all distances are integers).

Our first result concerns online algorithms with small recourse in the incremental setting; that is, when clients only arrive and not depart.

▶ **Theorem 1.** *There is a deterministic online algorithm for the facility location problem that achieves a competitive ratio of* $(1+\sqrt{2}+\epsilon)$ *with* $O\left(\frac{\log n}{\epsilon}\log\frac{1}{\epsilon}\right)$ *amortized facility and client recourse against an adaptive adversary.*

Our next result is in the online dynamic model with the goal to achieve small update time. First we consider incremental setting.

▶ **Theorem 2.** *In the incremental setting against an adaptive adversary, there is a randomized dynamic algorithm for the facility location problem that, with probability at least* $1-1/n^2$*, maintains an approximation factor of* $(1+\sqrt{2}+\epsilon)$ *with amortized update time of* $O(\frac{n}{\epsilon^2}\log^3 n\log\frac{1}{\epsilon})$*. Further, there exists an* $O(1)$*-approximate dynamic algorithm with* $O(\frac{|F|}{\epsilon^2}\log^3 n\log\frac{1}{\epsilon})$*-amortized update time.*

Note that it takes $\Theta(n|F|)$ space to specify the input in our model (see Section 2.1). Our algorithm for the above theorem also simultaneously achieves an amortized *logarithmic* client recourse per step; see the remark at the end of Section 5.

Our algorithms to show the above two theorems differ from the previous approaches used in the context of online variants of facility location problem, and is based on *local search*. The local search algorithm is one of the most widely used algorithms for the facility location problem in practice, and is known to achieve an approximation factor of $(1+\sqrt{2})$ in the offline setting. See the influential paper by Arya *et al* [2] and a survey by Munagala [25]. Thus our result matches the best known approximation ratio for offline facility location using local search. Further, our result shows that the local search algorithm augmented with some small modifications is *inherently stable* as it does not make too many changes to the solutions even if clients are added in an online fashion. This gives further justification for its popularity among practitioners.

Finally, we study the fully dynamic setting, where the clients arrive and depart. Here, we first consider an important class of metric spaces called hierarchically well separated tree (HST) metrics [3]; see Definition 13 for the formal definition, and Section 2.1 for more details about how the input sequence is given. For HST metric spaces, we show the following result.

▶ **Theorem 3.** *In the fully dynamic setting against adaptive adversaries, there is a deterministic algorithm for the facility location problem that achieves an* $O(1)$ *approximation factor with* $O(|F|)$ *preprocessing time and* $O(\log^3 D)$ *amortized update time for the HST metric spaces.*

A seminal result by Bartal [3], which was later tightened by Fakcharoenphol, Rao and Talwar [12], shows that any arbitrary N-point metric space can be embedded into a distribution over HSTs such that the expected distortion is at most $O(\log N)$, which is also tight. Moreover, such a probabilistic embedding can also be computed in $O(N^2\log N)$ time; see recent results by Blelloch, Gu and Sun for details [5]. These results immediately imply the following theorem, provided the input is specified as in Section 2.1.

▶ **Theorem 4.** *In the fully dynamic setting against oblivious adversary, there is a randomized algorithm for the facility location problem that maintains an approximation factor of* $O(\log|F|)$ *with preprocessing time of* $O(|F|^2\log|F|)$ *and* $O(\log^3 D)$ *amortized update time. The approximation guarantee holds only in expectation for every time step of the algorithm.*

Observe that unlike the incremental setting, the above theorem holds only in the oblivious adversary model, as probabilistic embedding techniques preserve distances only in expectation as can be seen by taking a cycle on n points. Our result also shows that probabilistic tree embeddings using HSTs can be a very useful technique in the design of dynamic algorithms, similar to its role in online algorithms [3, 4, 26, 6].

Our algorithms in Theorems 3 and 4 in the fully dynamic setting also have the property that amortized client and facility *recourse* is $O(\log^3 D)$ (in fact, we can achieve a slight better bound of $O(\log^2 D)$ as can be seen from the analysis). This holds as our dynamic algorithms maintain the entire assignment of clients to facilities *explicitly* in memory at every time step. Thus, the amortized client reconnections is at most the amortized update time.

1.2 Our Techniques

Our main algorithmic technique for proving Theorems 1 and 2 is local search, which is one of the powerful algorithm design paradigms. Indeed, for both results, the competitive (approximation) ratio we achieve is $1 + \sqrt{2} + \epsilon$, which matches the best approximation ratio for offline facility location obtained using local search [2]. Both of our results are based on the following key lemma. Suppose we maintain local optimum solutions at every time step in our algorithm. When a new client j_t comes at time t, we add it to our solution using a simple operation, and let Δ_t be the increase of our cost due to the arrival of j_t. The key lemma states that the sum of Δ_t values in the first T' time steps can be bounded in terms the optimum cost at time T'. With a simple modification to the local search algorithm, in which we require each local operation decreases enough cost for every client it reconnects, one can bound the total client recourse.

The straightforward way to implement the local search algorithm takes time $\Omega(n^3)$. To derive a better running time, we leverage the randomized local search idea of Charikar and Guha [8]. At every iteration, we randomly choose a facility i or a closing operation, and then perform the best operation that opens or swaps in i, or closes a facility if that is what we choose. By restricting the facility i and with the help of the heap data structure, an iteration of the algorithm can be implemented in time $O(|C| \log |F|)$. As in [8] we can also show that each iteration can make a reasonable progress in expectation, leading to a bound of $\tilde{O}(|F|)$ on the number of iterations for the success of the algorithm with high probability. We remark that the algorithm in [8] used a different local search framework. Therefore, our result shows that the classic algorithm of [2] can also be made fast.

However, directly replacing the randomized local search procedure with a deterministic one does not work: The solution at the end of each time might not be a local optimum as we did not enumerate all possible local operations. Thus the key lemma does not hold any more. Nevertheless we show that applying a few local operations around j_t upon its arrival can address the issue. With the key lemma, one can bound the number of times we perform the iterative randomized local search procedure, and thus the overall running time.

Our proof for Theorem 3 is based on a generalization of the greedy algorithm for facility location on HST metrics, which was developed in [11] in the context of *differential privacy* but only for the case of *uniform* facility cost. The intuition of the algorithm is as follows: If for some vertex v of the HST T, the number of clients in the tree T_v (the sub-tree of T rooted at v) times the length of parent edge of v is big compared to the cost of the cheapest facility in T_v, then we should open that facility. Otherwise, we should not open it and let the clients in T_v be connected to outside T_v through the parent edge. This intuition can be made formal: We mark v in the former case; then simply opening the cheapest facility in T_v for all *lowest marked* vertices v leads to a constant approximation for facility location.

The above offline algorithm leads to a *dynamic data structure* that maintains $O(1)$-approximate solutions, supports insertion and deletion of clients, and reports the connecting facility of a client in $O(\log D)$ time. This is the case since each time a client arrives or departs, only its ancestors will be affected. However, in a dynamic algorithm setting, we need to maintain the assignment vector in memory, so that when the connecting facility of a

client changes, it needs to be notified. This requires that the number of reconnections made by our algorithm to be small. To achieve the goal, we impose two constants for each v when deciding whether v should be marked and the cheapest facility in T_v should be open. When a vertex v changes its marking/opening status, we update the constants in such a way that it becomes hard for the status to be changed back.

1.3 More Related Work

Lattanzi and Vassilvitskii [22] first studied the role of recourse (which they call "consistency") in online k-clustering problem when clients only arrive – a setting similar to our Theorem 1. Lattanzi and Vassilvitskii gave a constant approximation algorithm with $O(k^2 \log^4 n)$ total recourse (reclusterings). They also showed an $\Omega(k \log n)$ recourse lowerbound. There are some subsequent works studying the k-clustering problems in the context of recourse and dynamic algorithms; we refer the readers to [22, 21, 7, 9] for more details.

A fully dynamic online model of facility location problem, where clients arrive and depart was recently studied by Cygan *et al.* [10], but with different assumption on recourse. In this model, the set of facility locations is the same as the set of clients and thus they are also changing dynamically. When a client arrives, the algorithm has to assign it to an open facility immediately; while upon departure of a client, if a facility was opened at the same location, then the clients that were assigned to that location should be reassigned immediately and irrevocably. They showed that a delicate extension of Meyerson's [24] algorithm obtains asymptotically tight competitive ratio of $O(\log n / \log \log n)$. However, the result holds only for the *uniform facility costs* and Cygan *et al* [10] even showed an *unbounded* lower bound for the non-uniform facility cost case in their model. Moreover, in their model reconnections of clients are assumed to be "automatic" and do not count towards the client recourse; it is not clear how many client reconnections their algorithm will make. Hence, our result in Theorem 4 also implies that one can circumvent the unbounded lowerbound for the general problem with small recourse.

As we noted earlier, a strictly online algorithm for facility location problem cannot achieve a competitive ratio better than $O(\log n / \log \log n)$. To overcome this, the problem has been studied in the so-called *incremental model*, which allows two open facilities to be merged at any given time. In this model [13, 17] showed that one can obtain $O(1)$ approximation for case when facility opening costs are uniform. However, the recourse of such an algorithm may be arbitrary.

2 Preliminaries

Throughout the paper, we use F to denote the set of potential facilities for all the problems and models; we assume F is given upfront. C is the dynamic set of clients we need to connect by our algorithm. This is not necessarily the set of clients that are present: In the algorithms for online facility location with recourse and dynamic facility location in the incremental setting, we fix the connections of some clients as the algorithms proceed. These clients are said to be "frozen" and excluded from C. We shall always use d to denote the hosting metric containing F and all potential clients. For any point j and subset V of points in the metric, we define $d(j, V) = \min_{v \in V} d(j, v)$ to be the minimum distance from j to a point in V. We assume all distances are integers, the minimum non-zero distance between two points is 1. We define D, the diameter or the aspect ratio of a metric space, as the largest distance between two points in it. Let n be $|F|$ plus the total number of clients arrived during the whole process. The algorithms do not need to know the exact value of

n in advance, except that in the dynamic algorithm for facility location in the incremental setting (the problem in Theorem 2), to achieve the $1 - 1/n^2$ success probability, a sufficiently large $\Gamma = \text{poly}(n, \log D, \frac{1}{\epsilon})$ needs to be given.[1]

In all the algorithms, we maintain a set S of open facilities, and a connection $\sigma \in S^C$ of clients in C to facilities in S. We do not require that σ connects clients to their respective nearest open facilities. For any solution $(S' \subseteq F, \sigma' \in S'^C)$, we use $\mathsf{cc}(\sigma') = \sum_{j \in C} d(j, \sigma_j)$ to denote the connection cost of the solution. For facility location, we use $\mathsf{cost}(S', \sigma') = f(S') + \mathsf{cc}(\sigma')$ to denote the total cost of the solution (S', σ'), where $f(S') := \sum_{i \in S'} f_i$. Notice that σ and the definitions of cc and cost functions depend on the dynamic set C.

Throughout the paper, we distinguish between a "moment", a "time" and a "step". A moment refers to a specific time point during the execution of our algorithm. A time corresponds to an arrival or a departure event: At each time, exactly one client arrives or departs, and time t refers to the period from the moment the t-th event happens until the moment the $(t+1)$-th event happens (or the end of the algorithm). One step refers to one statement in our pseudo-codes indexed by a number.

2.1 Specifying Input Sequence

In this section we specify how the input sequence is given. For the online and dynamic facility location problem, we assume the facility locations F, their costs $(f_i)_{i \in F}$, and the metric d restricted to F are given upfront, and they take $O(|F|^2)$ space. Whenever a client $j \in C$ arrives, it specifies its distance to every facility $i \in F$ (notice that the connection cost of an assignment $\sigma \in S^C$ does not depend on distances between two clients and thus they do not need to be given). Thus the whole input contains $O(n|F|)$ words.

For Theorems 3 and 4, as we do not try to optimize the constants, we *do not* need that a client specifies its distance to every facility. By losing a multiplicative factor of 2 and an additive factor of 1 in the approximation ratio, we can assume that every client j is collocated with its nearest facility in F Thus, we only require that when a client j comes, it reports the position of its nearest facility. For Theorem 3, the HST T over F is given at the beginning using $O(|F|)$ words. For Theorem 4, the metric d over F is given at the beginning using $O(|F|^2)$ words. Then, we use an efficient algorithm [5] to sample a HST T.

2.2 Local Search for facility location

The local-search technique has been used to obtain the classic $(1+\sqrt{2})$-approximation offline algorithm for facility location [2]. We now give an overview of the algorithm, which will be the baseline of our online and dynamic algorithms for facility location. One can obtain a (tight) 3-approximation for facility location without scaling facility costs. Scaling the facility costs by a factor of $\lambda := \sqrt{2}$ when deciding whether an operation can decrease the cost, we can achieve a better approximation ratio of $\alpha_{\mathsf{FL}} := 1 + \sqrt{2}$. Throughout, we fix the constants $\lambda = \sqrt{2}$ and $\alpha_{\mathsf{FL}} = 1 + \sqrt{2}$. For a solution (S', σ') to a facility location instance, we use $\mathsf{cost}_\lambda(S', \sigma') = \lambda f(S') + \mathsf{cc}(\sigma')$ to denote the cost of the solution (S', σ') with facility costs scaled by $\lambda = \sqrt{2}$. We call $\mathsf{cost}_\lambda(S', \sigma')$ the *scaled cost* of (S', σ').

Given the current solution (S, σ) for a facility location instance defined by F, C, d and $(f_i)_{i \in F}$, we can apply a *local operation* that changes the solution (S, σ). A valid local operation is one of the following.

[1] For an algorithm that might fail, we need to have some information about n to obtain a failure probability that depends on n.

- An open operation, in which we open some facility $i \in F$ and reconnect a subset $C' \subseteq C$ of clients to i. We allow i to be already in S, in which case we simply reconnect C' to i. This needs to be allowed since our σ does not connect clients to their nearest open facilities.
- A close operation, we close some facility $i' \in S$ and reconnect the clients in $\sigma^{-1}(i')$ to facilities in $S \setminus \{i'\}$.
- In a swap operation, we open some facility $i \notin S$ and close some facility $i' \in S$, reconnect the clients in $\sigma^{-1}(i')$ to facilities in $S \setminus \{i'\} \cup \{i\}$, and possibly some other clients to i. We say i is *swapped in* and i' is *swapped out* by the operation.

Thus, in any valid operation, we can open and/or close at most one facility. A client can be reconnected if it is currently connected to the facility that will be closed, or it will be connected to the new open facility. After we apply a local operation, S and σ will be updated accordingly so that (S, σ) is always the current solution.

For the online algorithm with recourse model, since we need to bound the number of reconnections, we apply a local operation only if the *scaled* cost it decreases is large compared to the number of reconnections it makes. This motivates the following definition:

▶ **Definition 5** (Efficient operations for facility location)**.** *Given a $\phi \geq 0$, we say a local operation on a solution (S, σ) for a facility location instance is ϕ-efficient, if it decreases $\mathsf{cost}_\lambda(S, \sigma)$ by more than ϕ times the number of clients it reconnects.*

The following theorem can be derived from the analysis for the local search algorithms for facility location.

▶ **Theorem 6.** *Consider a facility location instance with cost of the optimum solution being* opt *(using the original cost function). Let (S, σ) be the current solution in our algorithm and $\phi \geq 0$ be a real number. If there are no ϕ-efficient local operations on (S, σ), then we have*

$$\mathsf{cost}(S, \sigma) \leq \alpha_{\mathsf{FL}}\big(\mathsf{opt} + |C|\phi\big).$$

In particular, if we apply the theorem with $\phi = 0$, then we obtain that (S, σ) is an $(\alpha_{\mathsf{FL}} = 1 + \sqrt{2})$-approximation for the instance.

2.3 A Useful Lemma

The following lemma will be used repeatedly in our analysis. It's not hard to prove, but due to space constraints, we only include the proof in the full version of this paper.

▶ **Lemma 7.** *Let $b \in \mathbb{R}^T_{\geq 0}$ for some integer $T \geq 1$. Let $B_{T'} = \sum_{t=1}^{T'} b_t$ for every $T' = 0, 1, \cdots, T$. Let $0 < a_1 \leq a_2 \leq \cdots \leq a_T$ be a sequence of real numbers and $\alpha > 0$ such that $B_t \leq \alpha a_t$ for every $t \in [T]$. Then we have*

$$\sum_{t=1}^{T} \frac{b_t}{a_t} \leq \alpha \left(\ln \frac{a_T}{a_1} + 1 \right).$$

Organization. The rest of the paper is organized as follows. In Section 3, we prove Theorem 1 by giving our online algorithm for facility location with recourse. Section 4 gives the randomized local search procedure, that will be used in the proof of Theorem 2 in Section 5. We give some open problems and future directions in Section 6. Finally, the proof of Theorem 4 is deferred to Appendix A, where we give the fully dynamic algorithm for facility location in HST metrics. Due to space constraint, some proofs are not included in this paper, and we refer interested readers to the full version.

3 $(1+\sqrt{2}+\epsilon)$-Competitive Online Algorithm with Recourse

In this section, we prove Theorem 1 by giving the algorithm for online facility location with recourse.

3.1 The Algorithm

For any $\epsilon > 0$, let $\epsilon' = \Theta(\epsilon)$ be a parameter that is sufficiently small so that the approximation ratio $\alpha_{\mathsf{FL}} + O(\epsilon') = 1 + \sqrt{2} + O(\epsilon')$ achieved by our algorithm is at most $\alpha_{\mathsf{FL}} + \epsilon$. Our algorithm for online facility location is easy to describe. Whenever the client j_t comes at time t, we use a simple rule to connect j_t, as defined in the procedure initial-connect in Algorithm 1: either connecting j_t to the nearest facility in S, or opening and connecting j_t to its nearest facility in $F \setminus S$, whichever incurs the smaller cost. Then we repeatedly perform ϕ-efficient operations (Definition 5), until no such operations can be found, for $\phi = \frac{\epsilon' \cdot \mathsf{cost}(S,\sigma)}{\alpha_{\mathsf{FL}}|C|}$. [2]

Algorithm 1 initial-connect(j).

1: **if** $\min_{i \in F \setminus S}(f_i + d(i,j)) < d(j,S)$ **then**
2: let $i^* = \arg\min_{i \in F \setminus S}(f_i + d(i,j))$, $S \leftarrow S \cup \{i^*\}, \sigma_j \leftarrow i^*$
3: **else** $\sigma_j \leftarrow \arg\min_{i \in S} d(j,i)$

We can show that the algorithm gives an $(\alpha_{\mathsf{FL}} + \epsilon)$-approximation with amortized recourse $O(\log D \log n)$; recall that D is the aspect ratio of the metric. To remove the dependence on D, we divide the algorithm into stages, and *freeze* the connections of clients that arrived in early stages. The final algorithm is described in Algorithm 3, and Algorithm 2 gives one stage of the algorithm.

Algorithm 2 One Stage of Online Algorithm for Facility Location.

Input:
- C: initial set of clients
- (S,σ): a solution for C which is $O(1)$-approximate
- Clients $j_1, j_2, \cdots$ arrive from time to time

Output: Guaranteeing that (S,σ) at the end of each time t is $\frac{\alpha_{\mathsf{FL}}}{1-\epsilon'}$-approximate

1: init $\leftarrow \mathsf{cost}(S,\sigma)$
2: **for** $t \leftarrow 1, 2, \cdots$, terminating if no more clients will arrive **do**
3: $C \leftarrow C \cup \{j_t\}$, and call initial-connect(j_t)
4: **while** there exists an $\frac{\epsilon' \cdot \mathsf{cost}(S,\sigma)}{\alpha_{\mathsf{FL}}|C|}$-efficient local operation **do** perform the operation
5: **if** $\mathsf{cost}(S,\sigma) > \mathsf{init}/\epsilon'$ **then** terminate the stage

In Algorithm 2, we do as described above, with two modifications. First, we are given an initial set C of clients and a solution (S,σ) for C which is $O(1)$-approximate. Second, the stage will terminate if the cost of our solution increases by a factor of more than $1/\epsilon'$. The main algorithm (Algorithm 3) is broken into many stages. Since we shall focus on one stage of the algorithm for most part of our analysis, we simply redefine the time so that every stage starts with time 1. The improved recourse comes from the *freezing* operation: at the

[2] There are exponential number of possible operations, but we can check if there is a ϕ-efficient one efficiently. close operations can be handled easily. To check if we can open a facility i, it suffices to check if $\sum_{j \in C: d(j,i)+\phi < d(j,\sigma_j)} (d(j,\sigma_j) - d(j,i) - \phi) > \lambda f_i \cdot 1_{i \notin S}$. swap operations are more complicated but can be handled similarly.

Algorithm 3 Online Algorithm for Facility Location.

1: $C \leftarrow \emptyset, S \leftarrow \emptyset, \sigma = ()$
2: **repeat**
3: $C^\circ \leftarrow C, (S^\circ, \sigma^\circ) \leftarrow (S, \sigma)$
4: redefine the next time to be time 1 and run one stage as defined in Algorithm 2
5: permanently open one copy of each facility in S°, and permanently connect clients in C° according to σ° (we call the operation *freezing* S° and C°)
6: $C \leftarrow C \setminus C^\circ$, restrict the domain of σ to be the new C
7: **until** no clients come

end of each stage, we permanently open one copy of each facility in S°, and permanently connect clients in C° to copies of S° according to σ°, where C° and (S°, σ°) are the client set and solution at the beginning of the stage. Notice that we assume the original facilities in S° will still participate in the algorithm in the future; that is, they are subject to opening and closing. Thus each facility may be opened multiple times during the algorithm and we take the facility costs of all copies into consideration. This assumption is only for the sake of analysis; the actual algorithm only needs to open one copy and the costs can only be smaller compared to the described algorithm.

From now on, we focus on one stage of the algorithm and assume that the solution given at the beginning of each stage is $O(1)$-approximate. In the end we shall account for the loss due to the freezing of clients and facilities. Within a stage, the approximation ratio follows directly from Theorem 6: Focus on the moment after the while loop at time step t in Algorithm 2. Since there are no $\frac{\epsilon' \cdot \mathsf{cost}(S,\sigma)}{\alpha_{\mathsf{FL}}|C|}$-efficient local operations on (S, σ), we have by the theorem that $\mathsf{cost}(S,\sigma) \le \alpha_{\mathsf{FL}} \left(\mathsf{opt} + |C| \cdot \frac{\epsilon' \cdot \mathsf{cost}(S,\sigma)}{\alpha_{\mathsf{FL}}|C|} \right) = \alpha_{\mathsf{FL}}\mathsf{opt} + \epsilon' \cdot \mathsf{cost}(S,\sigma)$, where opt is the cost of the optimum solution for C. Thus, at the end of each time, we have $\mathsf{cost}(S,\sigma) \le \frac{\alpha_{\mathsf{FL}}}{1-\epsilon'} \cdot \mathsf{opt}$.

3.2 Bounding Amortized Recourse in One Stage

We then bound the amortized recourse in a stage; we assume that $\mathsf{cost}(S,\sigma) > 0$ at the beginning of the stage since otherwise there will be no recourse involved in the stage (since we terminate the stage when the cost becomes non-zero). We use T to denote the last time of the stage. For every time t, let C_t be the set C at the end of time t, and opt_t to be the cost of the optimum solution for the set C_t. For every $t \in [T]$, we define Δ_t to be the value of $\mathsf{cost}(S,\sigma)$ after Step 3 at time step t in Algorithm 2, minus that before Step 3. We can think of this as the cost increase due to the arrival of j_t.

The key lemma we can prove is the following:

▶ **Lemma 8.** *For every $T' \in [T]$, we have*

$$\sum_{t=1}^{T'} \Delta_t \le O(\log T')\mathsf{opt}_{T'}.$$

With Lemma 8, we can now bound the amortized recourse of one stage. In time t, $\mathsf{cost}(S,\delta)$ first increases by Δ_t in Step 3. Then after that, it decreases by at least $\frac{\epsilon'\mathsf{cost}(S,\sigma)}{\alpha_{\mathsf{FL}}|C|} \ge \frac{\epsilon'\mathsf{opt}_t}{\alpha_{\mathsf{FL}}|C|} \ge \frac{\epsilon'\mathsf{opt}_t}{\alpha_{\mathsf{FL}}|C_T|}$ for every reconnection we made. Let $\Phi_{T'} = \sum_{t=1}^{T'} \Delta_t$; Lemma 8 says $\Phi_t \le \alpha\mathsf{opt}_t$ for some $\alpha = O(\log T)$ and every $t \in [T]$. Noticing that $(\mathsf{opt}_t)_{t\in T}$ is a non-decreasing sequence, the total number of reconnections is at most

$$\frac{\mathsf{init}}{\epsilon' \cdot \mathsf{opt}_1/(\alpha_{\mathsf{FL}}|C_T|)} + \sum_{t=1}^{T} \frac{\Delta_t}{\epsilon' \cdot \mathsf{opt}_t/(\alpha_{\mathsf{FL}}|C_T|)} = \frac{\alpha_{\mathsf{FL}}|C_T|}{\epsilon'} \left(\frac{\mathsf{init}}{\mathsf{opt}_1} + \sum_{t=1}^{T-1} \frac{\Delta_t}{\mathsf{opt}_t} + \frac{\Delta_T}{\mathsf{opt}_T} \right).$$

Notice that $\mathsf{init} \le O(1)\mathsf{opt}_0 \le O(1)\mathsf{opt}_1$. Applying Lemma 7 with T replaced by $T-1$, $b_t = \Delta_t, B_t = \Phi_t$ and $a_t = \mathsf{opt}_t$ for every t, we have that $\sum_{t=1}^{T-1} \frac{\Delta_t}{\mathsf{opt}_t} \le \alpha \left(\ln \frac{\mathsf{opt}_{T-1}}{\mathsf{opt}_1} + 1 \right) = O\left(\log T \log \frac{1}{\epsilon'} \right)$, since we have $\mathsf{opt}_{T-1} \le O(1/\epsilon') \cdot \mathsf{opt}_1$. Notice that $\Delta_T \le \mathsf{opt}_T$ since $\mathsf{opt}_T \ge \min_{i \in F}(f_i + d(i, j_T)) \ge \Delta_T$. So, the total number of reconnections is at most $O\left(\frac{\log T}{\epsilon'} \log \frac{1}{\epsilon'} \right) \cdot |C_T|$. The amortized recourse per client is $O\left(\frac{\log T}{\epsilon'} \log \frac{1}{\epsilon'} \right) \le O\left(\frac{\log n}{\epsilon'} \log \frac{1}{\epsilon'} \right)$, where in the amortization, we only considered clients involved in the stage. Recall that n is the total number of clients arrived.

As each client appears in at most 2 stages, the overall amortized recourse is $O\left(\frac{\log n}{\epsilon'} \log \frac{1}{\epsilon'} \right)$. Finally we consider the loss in the approximation ratio due to freezing of clients. Suppose we are in the p-th stage. Then the clients arrived at and before $(p-2)$-th stage has been frozen and removed. Let $\overline{\mathsf{opt}}$ be the cost of the optimum solution for all clients arrived at or before $(p-1)$-th stage. Then the frozen facilities and clients have cost at most $\overline{\mathsf{opt}} \cdot O\left(\epsilon' + \epsilon'^2 + \epsilon'^2 + \cdots \right) = O(\epsilon')\overline{\mathsf{opt}}$. In any time in the p-th stage, the optimum solution taking all arrived clients into consideration has cost $\overline{\mathsf{opt}}' \ge \overline{\mathsf{opt}}$, and our solution has cost at most $(\alpha_{\mathsf{FL}} + O(\epsilon'))\overline{\mathsf{opt}}'$ without considering the frozen clients and facilities. Thus, our solution still has approximation ratio $\frac{(\alpha_{\mathsf{FL}} + O(\epsilon'))\overline{\mathsf{opt}}' + O(\epsilon')\overline{\mathsf{opt}}}{\overline{\mathsf{opt}}'} = \alpha_{\mathsf{FL}} + O(\epsilon')$ when taking the frozen clients into consideration.

4 Fast Local Search via Randomized Sampling

From now on, we will be concerned with dynamic algorithms. Towards proving Theorem 2 for the incremental setting, we first develop a randomized procedure that allows us to perform local search operations fast. In the next section, we use this procedure and ideas from the previous section to develop the dynamic algorithm with the fast update time.

The high level idea is as follows: We partition the set of local operations into many "categories" depending on which facility it tries to open or swap in. In each iteration of the procedure, we sample the category according to some distribution and find the best local operation in this category. By only focusing on one category, one iteration of the procedure can run in time $O(|C| \log |F|)$. On the other hand, the categories and the distribution over them are designed in such a way that in each iteration, the cost of our solution will be decreased by a multiplicative factor of $1 - \Omega\left(\frac{1}{|F|}\right)$. This idea has been used in [8] to obtain their $\tilde{O}(n^2)$ algorithm for approximating facility location. However, their algorithm was based on a different local search algorithm and analysis; for consistency and convenience of description, we stick to original local search algorithm of [2] that leads to $(1+\sqrt{2})$-approximation for the problem. Our algorithm needs to use the heap data structure.

4.1 Maintaining Heaps for Clients

Unlike the online algorithm for facility location in Section 3, in the dynamic algorithm, we guarantee that the clients are connected to their nearest open facilities. That is, we always have $\sigma_j = \arg\min_{i \in S} d(j, i)$; we still keep σ for convenience of description. We maintain $|C|$ min-heaps, one for each client $j \in C$: The min-heap for j will contain the facilities in

Algorithm 4 Δ-open(i).

1: **return** $\lambda f_i - \sum_{j \in C} \max\{0, d(j,\sigma_j) - d(j,i)\}$

Algorithm 5 try-open(i).

1: **if** Δ-open$(i) < 0$ **then** open i by updating S, σ and heaps accordingly

Algorithm 6 Δ-swap-in(i).

1: $C' \leftarrow \{j \in C : d(j,i) < d(j,\sigma_j)\}$ and $\Psi \leftarrow \lambda f_i - \sum_{j \in C'} \big(d(j,\sigma_j) - d(j,i)\big)$
2: $\Delta \leftarrow \min_{i' \in S} \left\{ \sum_{j \in \sigma^{-1}(i') \setminus C'} \left[\min\{d(j,i), d(j,\text{heap-top}(j))\} - d(j,i') \right] - \lambda f_{i'} \right\} + \Psi$
3: **return** $(\Delta,$ the i' above achieving the value of $\Delta)$

Algorithm 7 Δ-close.

1: $\Delta \leftarrow \min_{i' \in S} \left\{ \sum_{j \in \sigma^{-1}(i')} \left[d(j,\text{heap-top}(j)) - d(j,i') \right] - \lambda f_{i'} \right\}$
2: **return** $(\Delta,$ the i' above achieving the value of $\Delta)$

$S \setminus \{\sigma_j\}$, with priority value of i being $d(j,i)$. This allows us to efficiently retrieve the second nearest open facility to each j: This is the facility at the top of the heap for j and we use the procedure heap-top(j) to return it.

We define four simple procedures Δ-open, try-open, Δ-swap-in and Δ-close that are described in Algorithms 4, 5, 6 and 7 respectively. Recall that we use the *scaled cost* for the local search algorithm; so we are working on the scaled cost function in all these procedures. Δ-open(i) for any $i \notin S$ returns Δ, the increment of the scaled cost that will be incurred by opening i. (For it to be useful, Δ should be negative, in which case $|\Delta|$ indicates the cost decrement of opening i). This is just one line procedure as in Algorithm 4; try-open will open i if it can reduce the scaled cost. Δ-swap-in(i) for some $i \notin S$ returns a pair (Δ, i'), where Δ is the smallest scaled cost increment we can achieve by opening i and closing some facility $i' \in S$, and i' gives the facility achieving the smallest value. (Again, for Δ to be useful, it should be negative, in which case i' is the facility that gives the maximum scaled cost decrement $|\Delta|$.) Similarly, Δ-close returns a pair (Δ, i'), which tells us the maximum scaled cost decrement we can achieve by closing one facility and which facility can achieve the decrement. Notice that in all the procedures, the facility we shall open or swap in is given as a parameter, while the facility we shall close is chosen and returned by the procedures.

With the heaps, the procedures Δ-open, Δ-swap-in and Δ-close can run in $O(|C|)$ time. We only analyze Δ-swap-in(i) as the other two are easier. First, we define C' to be the set of clients j with $d(j,i) < d(j,\sigma_j)$; these are the clients that will surely be reconnected to i once i is swapped in. Let $\Psi = \lambda f_i - \sum_{j \in C'} (d(j,\sigma_j) - d(j,i))$ be the net scaled cost increase by opening i and connecting C' to i. The computation of C' and Ψ in Step 1 takes $O(|C|)$ time. If additionally we close some $i' \in S$, we need to reconnect each client in $\sigma^{-1}(i') \setminus C'$ to either i, or the top element in the heap for j, whichever is closer to j. Steps 2 and 3 compute and return the best scaled cost increment and the best i'. Since $\sum_{i' \in S} |\sigma^{-1}(i')| = |C|$, the running time of the step can be bounded by $O(|C|)$.

The running time for try-open, swapping two facilities and closing a facility (which are not defined explicitly as procedures, but used in Algorithms 8) can be bounded by $O(|C| \log |F|)$. The running times come from updating the heap structures: For each of the $|C|$ heaps, we need to delete and/or add at most 2 elements; each operation takes time $O(\log |F|)$.

Algorithm 8 sampled-local-search.

```
1: if rand(0,1) < 1/3 then        ▷ rand(0,1) returns a uniformly random number in [0,1]
2:     (Δ, i') ← Δ-close
3:     if Δ < 0 then close i' by updating S, δ and heaps accordingly
4: else
5:     i ← random facility in F \ S
6:     Δ ← Δ-open(i), (Δ', i') ← Δ-swap-in(i)
7:     if Δ ≤ Δ' and Δ < 0 then open i by updating S, δ and heaps accordingly
8:     else if Δ' < 0 then open i and close i' by updating S, δ and heaps accordingly
```

Algorithm 9 FL-iterate(M).

```
1: (S^best, σ^best) ← (S, σ)
2: for ℓ ← 1 to M do
3:     call sampled-local-search
4:     if cost(S, σ) < cost(S^best, σ^best) then (S^best, σ^best) ← (S, σ)
5: return (S^best, σ^best)
```

4.2 Random Sampling of Local Operations

With the support of the heaps, we can design a fast algorithm to implement randomized local search. sampled-local-search in Algorithm 8 gives one iteration of the local search. We first decide which operation we shall perform randomly. With probability $1/3$, we perform the close operation that will reduce the scaled cost the most (if it exists). With the remaining probability $2/3$, we perform either an open or a swap operation. To reduce the running time, we randomly choose a facility $i \in F \setminus S$ and find the best operation that opens or swaps in i, and perform the operation if it reduces the cost. One iteration of sampled-local-search calls the procedures in Algorithms 4 to 7 at most once and performs at most one operation, and thus has running time $O(|C| \log |F|)$.

In the procedure FL-iterate(M) described in Algorithm 9, we run the sampled-local-search M times. It returns the best solution obtained in these iterations, according to the *original (non-scaled) cost*, which is not necessarily the solution given in the last iteration. So we have

▶ **Observation 9.** *The running time of* FL-iterate(M) *is* $O(M|C| \log |F|)$*, where* C *is the set of clients when we run the procedure.*

Throughout this section, we fix a facility location instance. Let (S^*, σ^*) be the optimum solution (w.r.t the original cost) and $\mathsf{opt} = \mathsf{cost}(S^*, \sigma^*)$ be the optimum cost. Fixing one execution of sampled-local-search, we use (S^0, σ^0) to denote the solution before the execution. Then, we have

▶ **Lemma 10.** *Let* (S°, σ°) *be the* (S, σ) *at the beginning of an execution of* FL-iterate(M)*, and assume it is an* $O(1)$*-approximation to the instance. Let* $\Gamma \geq 2$ *and* $M = O\left(\frac{|F|}{\epsilon'} \log \Gamma\right)$ *is big enough. Then with probability at least* $1 - \frac{1}{\Gamma}$*, the solution returned by the procedure is* $(\alpha_{\mathsf{FL}} + \epsilon')$*-approximate.*

5 $(1+\sqrt{2}+\epsilon)$-Approximate Dynamic Algorithm for Facility Location in Incremental Setting

In this section, we prove Theorem 2 by combining the ideas from Sections 3 and 4 to derive a dynamic algorithm for facility location in the incremental setting. As for the online algorithm in Section 3, we divide our algorithm into stages. Whenever a client comes, we use a simple rule to accommodate it. Now we can not afford to consider all possible local operations as in Section 3. Instead we use the randomized local search idea from the algorithm in Section 4 by calling the procedure FL-iterate. We call the procedure only if the cost of our solution has increased by a factor of $1+\epsilon'$ (where $\epsilon'=\Theta(\epsilon)$ is small enough). In our analysis, we show a lemma similar to Lemma 8: The total increase of costs due to arrival of clients is small, compared to the optimum cost for these clients. Then, we can bound the number of times we call FL-iterate. Recall that we are given an integer $\Gamma=\text{poly}\left(n,\log D,\frac{1}{\epsilon}\right)$ that is big enough: We are aiming at a success probability of $1-1/\Gamma$ for each call of FL-iterate. Our final running time will only depend on $O(\log\Gamma)$.

The main algorithm will be the same as Algorithm 3, except that we use Algorithm 10 as the algorithm for one stage. As before, we only need to design one stage of the algorithm. Recall that in a stage we are given an initial set C of clients, an $O(1)$-approximate solution (S,σ) for C. Clients come one by one and our goal is to maintain an $(\alpha_{\mathsf{FL}}+O(\epsilon'))$-approximate solution at any time. The stage terminates if no client comes or our solution has cost more than $1/\epsilon'$ times the cost of the initial solution.

Algorithm 10 One Stage of Dynamic Algorithm for Facility Location.

Input: C: the initial set of clients
(S,σ): initial solution for C, which is $O(1)$-approximate

1: let $M=O\left(\frac{|F|}{\epsilon'}\log\Gamma\right)$ be large enough
2: $(S,\sigma)\leftarrow$ FL-iterate (M), init $\leftarrow$ cost(S,σ), last $\leftarrow$ init
3: **for** $t\leftarrow 1,2,3,\cdots$, terminating if no more clients arrive **do**
4: **for** $q=\left\lceil\log\frac{\text{last}}{|F|}\right\rceil$ to $\left\lceil\log\frac{\text{last}}{\epsilon'}\right\rceil$ **do**
5: **if** $i\leftarrow\arg\min_{i\in F\setminus S,f_i\le 2^q}d(j_t,i)$ exists, **then** call try-open$'(i)$ ▷ try-open$'$ is the same as try-open except we consider the cost instead of scaled cost.
6: $C\leftarrow C\cup\{j_t\}$ and call try-open$'\left(\arg\min_{i\in F\setminus S}(d(j_t,i)+f_i)\right)$
7: **if** cost$(S,\sigma)>(1+\epsilon')\cdot$ last **then**
8: $(S,\sigma)\leftarrow$ FL-iterate (M)
9: **if** cost$(S,\sigma)>$ last **then** last $\leftarrow$ cost(S,σ)
10: **if** last $>$ init$/\epsilon'$ **then** terminate the stage

Notice that in a stage, we are considering the original costs of solutions (instead of scaled costs as inside FL-iterate). During a stage we maintain a value last which gives an estimation on the cost of the current solution (S,σ). Whenever a client j_t comes, we apply some rules to open some facilities and connect j_t (Steps 4 to 6). These operations are needed to make the cost increase due to the arrival of j_t (defined as Δ_t later) small. In the algorithm try-open$'$ is the same as try-open, except that we use the original cost instead of the scaled cost (this is not important but only for the sake of convenience). If cost(S,σ) becomes too large, i.e, cost$(S,\sigma)>(1+\epsilon')$last, then we call $(S,\sigma)\leftarrow$ FL-iterate(M) for the M defined in Step 1 (Step 8), and update last to cost(S,σ) if we have cost$(S,\sigma)>$ last (Step 9). We terminate the algorithm when last $\ge$ init$/\epsilon$, where init is cost(S,σ) at the beginning of the stage (Step 10).

We say an execution of FL-iterate(M) is successful if the event in Lemma 10 happens. Then we have

▶ **Lemma 11.** *If all executions of* FL-iterate *are successful, the solution* (S, σ) *at the end of each time is* $(1+\epsilon')(\alpha_{\mathsf{FL}}+\epsilon')$*-approximate.*

Now we argue each execution of FL-iterate(M) is successful with probability at least $1 - 1/\Gamma$. This will happen if (S, σ) is $O(1)$-approximate before the call. By Lemma 10, we only need to make sure that the (S, σ) before the execution is $O(1)$-approximate. This is easy to see: Before Step 6 in time t, we have $\mathsf{cost}(S, \sigma) \le O(1)\mathsf{opt}$; the increase of $\mathsf{cost}(S, \sigma)$ in the step is at most the value of opt after the step (i.e, we consider the client j_t when defining opt). Thus, we have $\mathsf{cost}(S, \sigma) \le O(1)\mathsf{opt}$ after the step.

5.1 Bounding Number of Times of Calling FL-iterate

It remains to bound the number of times we call FL-iterate. Again, we use T to denote the last time step of Algorithm 10 (i.e, one stage of the dynamic algorithm) and Δ_t to denote the cost increase due to the arrival of j_t: it is the value of $\mathsf{cost}(S, \sigma)$ before Step 6 minus that after Step 6 in time t. For every time $t \in [T]$, let C_t be the set C at the end of time t, and let opt_t be the cost of the optimum solution for C_t. Let last_t be the value of last at the *beginning* of time t.

Like in the online setting, we can bound the total cost increase in a similar manner as Lemma 8:

▶ **Lemma 12.** *For every* $T' \in [T-1]$*, we have*

$$\sum_{t=1}^{T'} \Delta_t \le O(\log T') \cdot \mathsf{opt}_{T'}$$

Then we can wrap up the proof of Theorem 2. Between two consecutive calls of FL-iterate in Step 8 at time t_1 and $t_2 > t_1$, $\mathsf{cost}(S, \sigma)$ should have increased by at least $\epsilon'\mathsf{last}_{t_2}$: At the end of time t_1, we have $\mathsf{cost}(S, \sigma) \le \mathsf{last}_{t_1+1} = \mathsf{last}_{t_2}$ since otherwise last should have been updated in time t_1. We need to have $\mathsf{cost}(S, \sigma) > (1+\epsilon')\mathsf{last}_{t_2}$ after Step 6 at time t_2 in order to call FL-iterate. Thus, the increase of the cost during this period is at least $\epsilon'\mathsf{last}_{t_2}$. Thus, we have $\sum_{t=t_1+1}^{t_2} \frac{\Delta_t}{\epsilon' \cdot \mathsf{last}_t} \ge 1$ since $\mathsf{last}_t = \mathsf{last}_{t_2}$ for every $t \in (t_1, t_2]$. The argument also holds when $t_1 = 0$ and $t_2 > t_1$ is the first time in which we call FL-iterate. Counting the call of FL-iterate in Step 2, we can bound the total number of times we call the procedure by $1 + \frac{1}{\epsilon'} \sum_{t=1}^{T} \frac{\Delta_t}{\mathsf{last}_t}$.

Again let $\Phi_{T'} = \sum_{t=1}^{T'} \Delta_t$ for every $T' \in [0, T]$. Lemma 12 says $\Phi_t \le O(\log t)\mathsf{opt}_t$ for every $t \in [0, T-1]$. For every $t \in [T]$, since $\Delta_t \le \mathsf{opt}_t$, thus we have $\Phi_t = \Phi_{t-1} + \Delta_t \le O(\log t)\mathsf{opt}_{t-1} \le O(\log T)\mathsf{last}_t$ since last_t will be at least the cost of some solution for C_{t-1}. Applying Lemma 7 with $a_t = \mathsf{last}_t, b_t = \Delta_t$ and $B_t = \Phi_t$ for every t, the number of times we call FL-iterate can be bounded by

$$1 + \frac{1}{\epsilon'} \sum_{t=1}^{T} \frac{\Delta_t}{\mathsf{last}_t} \le \frac{1}{\epsilon'} O(\log T) \left(\ln \frac{\mathsf{last}_T}{\mathsf{last}_1} + 1 \right) = O\left(\frac{\log T}{\epsilon} \log \frac{1}{\epsilon} \right).$$

We can then analyze the running time and the success probability of our algorithm. Focus on each stage of the algorithm. By Observation 9, each call to FL-iterate(M) takes time $O(M|C| \log |F|) = O\left(\frac{|F|}{\epsilon'} (\log \Gamma) |C| \log n \right) = O\left(\frac{n \cdot |C_T|}{\epsilon} \log^2 n \right)$, where C is the set of clients

in the algorithm at the time we call the procedure, $C_T \supseteq C$ is the number set of clients at the end of time T, and $M = O\left(\frac{|F|}{\epsilon'} \log \Gamma\right)$ is as defined in Step 1. The total number of times we call the procedure is at most $O\left(\frac{\log T}{\epsilon} \log \frac{1}{\epsilon}\right) \leq O\left(\frac{\log n}{\epsilon} \log \frac{1}{\epsilon}\right)$. Thus, the running time we spent on FL-iterate is $O\left(\frac{n \cdot |C_T|}{\epsilon^2} \log^3 n \log \frac{1}{\epsilon}\right)$. The running time for Steps 4 to 6 is at most $T \cdot O\big(\log \frac{|F|}{\epsilon'}\big) \cdot O\big(|C_T| \log |F|\big) = O(|C_T| T \log^2 \frac{|F|}{\epsilon}) \leq O(n|C_T| \log^2 \frac{n}{\epsilon})$. Thus, the total running time of a stage is at most $O\left(\frac{n \cdot |C_T|}{\epsilon^2} \log^3 n \log \frac{1}{\epsilon}\right)$. Now consider all the stages together. The sum of $|C_T|$ values over all stages is at most $2n$ since every client appears in at most 2 stages. So, the total running time of our algorithm is $O\left(\frac{n^2}{\epsilon^2} \log^3 n \log \frac{1}{\epsilon}\right)$.

For the success probability, we call FL-iterate(M) at most $O\left(\log_{1/\epsilon}(nD) \frac{\log n}{\epsilon} \log \frac{1}{\epsilon}\right)$ $= \text{poly}(\log n, \log D, \frac{1}{\epsilon})$ times. If we have Γ be at least n^2 times this number, which is still $\text{poly}(n, \log D, \frac{1}{\epsilon})$, then the success probability of our algorithm is at least $1 - 1/n^2$.

Finally, we remark that the success of the algorithm only depends on the success of all executions of FL-iterate. Each execution has success probability $1 - 1/\Gamma$ even if the adversary is adaptive. This finishes the proof of Theorem 2.

▶ Remark. We can indeed obtain an algorithm that has both $O(\log T)$ amortized client recourse and $\tilde{O}(n^2)$ running time, by defining $\phi = \frac{\text{cost}(S,\sigma)}{\alpha_{\text{FL}} \epsilon'}$ and only performing ϕ-efficient local operations. However, this will require us to put ϕ everywhere in our analysis and deteriorate the cleanness of the analysis. Thus, we choose to separate the two features in two algorithms: small recourse and $\tilde{O}(n^2)$ total running time.

We also remark that the total running time for all calls of FL-iterate is only $\tilde{O}(n|F|)$, and the $\tilde{O}(n^2)$ time comes from Steps 4 to 6. By losing a multiplicative factor of 2 and additive factor of 1 in the approximation ratio, we can assume every client is collocated with its nearest facility. Then at any time we only have $O(|F|)$ different positions for clients, and the running time of the algorithm can be improved to $O(\frac{n|F|}{\epsilon^2} \log^3 n \log \frac{1}{\epsilon})$.

6 Open Problems and Discussions

We initiated the study of facility location problem in general metric spaces in recourse and dynamic models. Several interesting problems remain open: The most obvious one is can we get $O(1)$-competitive online/dynamic algorithms with polylog amortized recourse or fast update times in the fully dynamic setting. Another interesting direction is can we extend our results to the capacitated facility location and capacitated k-median, where there is an upper bound on the number of clients that can be assigned to a single open facility. From technical point of view, it would be interesting to find more applications of local search and probabilistic tree embedding techniques in the dynamic algorithms model. Finally, as alluded in the introduction, an exciting research direction is to understand the power of recourse in the online model.

References

1 Aris Anagnostopoulos, Russell Bent, Eli Upfal, and Pascal Van Hentenryck. A simple and deterministic competitive algorithm for online facility location. *Inf. Comput.*, 194(2):175–202, November 2004. doi:10.1016/j.ic.2004.06.002.

2 Vijay Arya, Naveen Garg, Rohit Khandekar, Adam Meyerson, Kamesh Munagala, and Vinayaka Pandit. Local search heuristic for k-median and facility location problems. In *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 21–29, 2001. doi:10.1145/380752.380755.

3 Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 184–, Washington, DC, USA, 1996. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=874062.875536.

4 Yair Bartal, Avrim Blum, Carl Burch, and Andrew Tomkins. A polylog(n)-competitive algorithm for metrical task systems. In *Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997*, pages 711–719, 1997. doi:10.1145/258533.258667.

5 Guy E. Blelloch, Yan Gu, and Yihan Sun. Efficient construction of probabilistic tree embeddings. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, pages 26:1–26:14, 2017. doi:10.4230/LIPIcs.ICALP.2017.26.

6 Sébastien Bubeck, Michael B. Cohen, Yin Tat Lee, James R. Lee, and Aleksander Madry. k-server via multiscale entropic regularization. In *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, 2018. doi:10.1145/3188745.3188798.

7 T-H. Hubert Chan, Arnaud Guerqin, and Mauro Sozio. Fully dynamic k-center clustering. In *Proceedings of the 2018 World Wide Web Conference*, WWW ?18, page 579?587, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. doi:10.1145/3178876.3186124.

8 Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM J. Comput.*, 34(4):803–824, April 2005. doi:10.1137/S0097539701398594.

9 Vincent Cohen-Addad, Niklas Hjuler, Nikos Parotsidis, David Saulpic, and Chris Schwiegelshohn. Fully dynamic consistent facility location. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 3250–3260, 2019. URL: http://papers.nips.cc/paper/8588-fully-dynamic-consistent-facility-location.

10 Marek Cygan, Artur Czumaj, Marcin Mucha, and Piotr Sankowski. Online facility location with deletions. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 21:1–21:15, 2018. doi:10.4230/LIPIcs.ESA.2018.21.

11 Yunus Esencayi, Marco Gaboardi, Shi Li, and Di Wang. Facility location problem in differential privacy model revisited. *CoRR*, abs/1910.12050, 2019. arXiv:1910.12050.

12 Jittat Fakcharoenphol, Satish Rao, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In *Proceedings of the Thirty-fifth Annual ACM Symposium on Theory of Computing*, STOC '03, pages 448–455, New York, NY, USA, 2003. ACM. doi:10.1145/780542.780608.

13 Dimitris Fotakis. Incremental algorithms for facility location and k-median. *Theor. Comput. Sci.*, 361(2-3):275–313, 2006. doi:10.1016/j.tcs.2006.05.015.

14 Dimitris Fotakis. A primal-dual algorithm for online non-uniform facility location. *J. of Discrete Algorithms*, 5(1):141–148, March 2007. doi:10.1016/j.jda.2006.03.001.

15 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008. doi:10.1007/s00453-007-9049-y.

16 Dimitris Fotakis. Memoryless facility location in one pass. *ACM Trans. Algorithms*, 7(4):49:1–49:24, 2011. doi:10.1145/2000807.2000817.

17 Dimitris Fotakis. Online and incremental algorithms for facility location. *SIGACT News*, 42(1):97–131, March 2011. doi:10.1145/1959045.1959065.

18 Monia Ghobadi, Ratul Mahajan, Amar Phanishayee, Nikhil R. Devanur, Janardhan Kulkarni, Gireeja Ranade, Pierre-Alexandre Blanche, Houman Rastegarfar, Madeleine Glick, and Daniel C. Kilper. Projector: Agile reconfigurable data center interconnect. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*, pages 216–229, 2016. doi:10.1145/2934872.2934911.

19 Gramoz Goranci, Monika Henzinger, and Dariusz Leniowski. A tree structure for dynamic facility location. In *26th Annual European Symposium on Algorithms, ESA 2018, August 20-22, 2018, Helsinki, Finland*, pages 39:1–39:13, 2018. doi:10.4230/LIPIcs.ESA.2018.39.

20 Sudipto Guha and Samir Khuller. Greedy strikes back: Improved facility location algorithms. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '98, pages 649–657, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics. URL: http://dl.acm.org/citation.cfm?id=314613.315037.

21 Monika Henzinger, Dariusz Leniowski, and Claire Mathieu. Dynamic clustering to minimize the sum of radii. In Kirk Pruhs and Christian Sohler, editors, *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017, Vienna, Austria*, volume 87 of *LIPIcs*, pages 48:1–48:10. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017. doi:10.4230/LIPIcs.ESA.2017.48.

22 Silvio Lattanzi and Sergei Vassilvitskii. Consistent k-clustering. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1975–1984. PMLR, 2017. URL: http://proceedings.mlr.press/v70/lattanzi17a.html.

23 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. *Inf. Comput.*, 222:45–58, 2013.

24 A. Meyerson. Online facility location. In *Proceedings of the 42Nd IEEE Symposium on Foundations of Computer Science*, FOCS '01, pages 426–, Washington, DC, USA, 2001. IEEE Computer Society. URL: http://dl.acm.org/citation.cfm?id=874063.875567.

25 Kamesh Munagala. Local search for k-medians and facility location. In *Encyclopedia of Algorithms*, pages 1139–1143. Springer, 2016. doi:10.1007/978-1-4939-2864-4_212.

26 Seeun Umboh. Online network design algorithms via hierarchical decompositions. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1373–1387, 2015. doi:10.1137/1.9781611973730.91.

27 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, New York, NY, USA, 1st edition, 2011.

A Fully Dynamic Algorithm for Facility Location on Hierarchically Well Separated Tree Metrics

In this section, we give our fully dynamic algorithm for facility location on hierarchically-well-separated-tree (HST) metrics. Our algorithm achieves $O(1)$-approximation and $O(\log^2 D)$ amortized update time.

A.1 Hierarchically Well Separated Trees

▶ **Definition 13.** *A hierarchically-well-separated tree (or HST for short) is an edge-weighted rooted tree with the following properties:*

- *all the root-to-leaf paths have the same number of edges,*
- *if we define the level of vertex v, $\mathsf{level}(v)$, to be the number of edges in a path from v to any of its leaf descendant, then for an non-root vertex v, the weight of the edge between v and its parent is exactly $2^{\mathsf{level}(v)}$.*

Given a HST T with the set of leaves being X, we use d_T to denote the shortest path metric of the tree T (with respect to the edge weights) restricted to X.

The classic results by Bartal [3] and Fakcharoenphol, Rao and Talwar [12] state that we can embed any N-point metric (X, d) (with minimum non-zero distance being 1) to a distribution π of *expanding*[3] HST metrics (X, d_T) with distortion $O(\log N)$: For every $u, v \in X$, we have $d_T(u, v) \geq d(u, v)$ and $\mathbb{E}_{u,v}[d_T(u, v)] \leq O(\log N)d(u, v)$. Moreover, there is an efficient randomized algorithm [5] that outputs a sample of the tree T from π. Thus applying standard arguments, Theorem 3 implies Theorem 4.

Notations. As we mentioned early, we assume each client is collocated with a facility. From now on, we fix the HST T and assume the leaves of T is $X = F$; let V be the set of all nodes in T. Let d_T be the metric induced by T over the set V of vertices. Recall that $\mathsf{level}(v)$ is the level of v in T. For every vertex $v \in V$, define Λ_v to be the set of children of v, X_v to be the set of leaf descendants of v, and T_v be the maximal sub-tree of T rooted at v. We extend the facility cost from X to all vertices in V: for every $v \in V \setminus X$, we define $f_v = \min_{i \in X_v} f_i$. We can assume that each internal vertex v is a facility; by opening v we mean opening a copy of the $i \in X_v$ with $f_i = f_v$. This assumption only loses a factor of 2 in the competitive ratio: On one hand, having more facilities can only make our problem easier; on the other hand, the cost of connecting a client to any $i \in X_v$ is at most twice that of connecting it to v. By the definition, the facility costs along a root-to-leaf path are non-decreasing.

A.2 Offline Algorithm for Facility Location on HST Metrics

In this section, we first give an offline $O(1)$-approximation algorithm for facility location on the HST metric d_T as a baseline. Notice that facility location on trees can be solved exactly using dynamic programming. However the algorithm is hard to analyze in the dynamic algorithm model since the solution is sensitive to client arrivals and departures. Our algorithm generalizes the algorithm in [11] for facility location with uniform facility cost, that was used to achieve the differential privacy requirement.

For every vertex $v \in V$, we let N_v be the number of clients at locations in X_v. Although according to the definition N_v's are integers, in most part of the analysis we assume there are non-negative *real numbers*. This will be useful when we design the dynamic algorithm. Let $\alpha \in \{1, 2\}^V$ and $\beta \in \{1, 2\}^{V \setminus X}$ be vectors given to our algorithm. They are introduced solely for the purpose of extending the algorithm to the dynamic setting; for the offline algorithm we can set α and β to be all-1 vectors.

Marked and Open Facilities. For every vertex $v \in V$, we say v is *marked* w.r.t the vectors N and α if

$$N_v \cdot 2^{\mathsf{level}(v)} > f_v/\alpha_v$$

and *unmarked* otherwise. The following observation can be made:

▶ **Observation 14.** *Let u be the parent of v. If v is marked w.r.t N and α, so is u.*

Proof. v is marked w.r.t N and α implies $N_v 2^{\mathsf{level}(v)} > f_v/\alpha_v$. Notice that $N_u \geq N_v, \mathsf{level}(u) = \mathsf{level}(v) + 1, \alpha_v \leq 2\alpha_u$ and $f_u \leq f_v$. So, $N_u 2^{\mathsf{level}(u)} \geq 2N_v 2^{\mathsf{level}(v)} > 2f_v/\alpha_v \geq f_u/\alpha_u$. ◀

Thus there is a monotonicity property on the marking status of vertices in T. We say a vertex v is highest unmarked (w.r.t N and α) if it is unmarked and its parent is marked; we say a vertex v is lowest marked if it is marked but all its children are unmarked. However,

[3] A metric (X, d_T) is expanding w.r.t (X, d) if for every $u, v \in X$, we have $d_T(u, v) \geq d(u, v)$.

sometimes we say a vertex u is the lowest marked ancestor of a leaf $v \in X$ if either $u = v$ is marked, or $u \neq v$ is marked and the child of u in the u-v path is unmarked; notice that in this case, u might not be a lowest marked vertex since it may have some other marked children. If we need to distinguish between the two cases, we shall use that u is lowest marked *globally* to mean u is a lowest marked vertex.

If a leaf vertex $v \in X$ is marked, then we open v. For every marked vertex $v \in V \setminus X$, we open v if and only if

$$\left(\sum_{u \in \Lambda_v : u \text{ unmarked}} N_u \right) 2^{\mathsf{level}(v)} > f_v/(\alpha_v \beta_v).$$

Notice that all unmarked vertices are closed.

▶ **Observation 15.** *If v is lowest marked, then v is open.*

Proof. We can assume $v \notin X$ since otherwise v is open. So, $N_v 2^{\mathsf{level}(v)} > f_v/\alpha_v$ and all children of v are unmarked. Thus, $\sum_{u \in \Lambda_v : u \text{ unmarked}} N_u = \sum_{u \in \Lambda_v} N_u = N_v$. Therefore, $\left(\sum_{u \in \Lambda_v : u \text{ unmarked}} N_u\right) 2^{\mathsf{level}(v)} = N_v 2^{\mathsf{level}(v)} > f_v/\alpha_v \geq f_v/(\alpha_v \beta_v)$. Thus v will be open. ◀

With the set of open facilities defined, every client is connected to its nearest open facility according to d_T, using a consistent tie-breaking rule (e.g, the nearest open facility with the smallest index). We assume the root r of T has $\frac{f_v}{2^{\mathsf{level}(v)}} \leq 1$ by increasing the number of levels. So r will be marked whenever $N_r \geq 1$. This finishes the description of the offline algorithm.

Analysis of $O(1)$-Approximation Ratio. We show the algorithm achieves an $O(1)$-approximation. First we give a lower bound on the optimum cost. For every $v \in V$, let

$$\mathsf{LB}(v) = \min \left\{ N_v 2^{\mathsf{level}(v)}, f_v \right\}.$$

Then we have

▶ **Lemma 16.** *Let U be a set of vertices in T without an ancestor-descendant pair; i.e, for every two distinct vertex u and v in U, u is not an ancestor of v. Then the cost of the optimum solution is at least $\sum_{v \in U} \mathsf{LB}(v)$.*

Then let U be the set of highest unmarked vertices and marked leaves; clearly U does not have an ancestor-descendant pair. By Lemma 16, the optimum cost is at least $\sum_{v \in U} \mathsf{LB}(v)$. We prove the following lemma.

▶ **Lemma 17.** *The solution produced by our algorithm has cost at most $O(1) \sum_{u \in U} \mathsf{LB}(u)$.*

Combining Lemmas 16 and 17 gives that our algorithm is an $O(1)$-approximation. One lemma that will be useful in the analysis of dynamic algorithm is the following:

▶ **Lemma 18.** *For any open facility v in our solution, the number of clients connected to v that are outside T_v is at most $O(\log D) \frac{f_v}{2^{\mathsf{level}(v)}}$.*

▶ Remark. The algorithm so far gives a *data structure* that supports the following operations in $O(\log D)$ time: i) updating N_v for some $v \in X$ and ii) returning the nearest open facility of a leaf $v \in X$. Indeed the algorithm can be made simpler: We set α to be the all-1 vector, and we open the set of lowest marked facilities (so both α and β are not needed). For every vertex $u \in V$, we maintain the nearest open facility ψ_u to u in T_u. Whenever a client at v arrives or departs, we only need change N_u, ψ_u, marking and opening status of u for

ancestors u of v. To return the closest open facility to a leaf $v \in X$, we travel up the tree from v until we find an ancestor u with ψ_u defined, and return ψ_u. Both operations take $O(\log D)$ time. However, our goal is to maintain the solution (S, σ) *explicitly* in memory. Thus we also have to bound the the number of reconnections during the algorithm, since that will be a lower bound on the total running time.

A.3 Dynamic Algorithm for Facility Location on HST Metrics

In this section, we extend the offline algorithm to a dynamic algorithm with $O(\log^3 D)$-amortized update time; recall that D is the aspect ratio of the metric. We maintain α, β and N-vectors, and at any moment of the algorithm, the marking and opening status of vertices are exactly the same as that obtained from the offline algorithm for α, β and N.

Initially, let α and β be all-1 vectors, and N be the all-0 vector. So all the vertices are unmarked. Whenever a client at some $v \in X$ arrives or departs, the α, β values, the marking and opening status of ancestors of v may change and we show how to handle the changes. The vertices that are not ancestors of v are not affected during the process.

When a client at v arrives or departs, we increase or decrease the N_u values for all ancestors u of v by 1 *continuously* at the same rate (we can think of that the number of clients at v increases or decreases by 1 continuously). During the process, the marking and opening status of these vertices may change. If such an event happens, we change α and/or β values of the vertex so that it becomes harder for the status to change back in the future. Specifically, we use the following rules:

- If a vertex u changes to marked (from being unmarked), then we change α_u to 2 (notice that u remains marked w.r.t the new α), and β_u to 1. In this case, we do not consider the opening status change of u as an event.
- If a vertex u changes to unmarked (from being marked), we change α_u to 1 (notice that u remains unmarked w.r.t the new α). The β_u value becomes useless. In this case, we also do not consider the opening status change of u as an event.
- If a marked vertex u becomes open (from being closed), then we change β_u to 2 (notice that u remains open w.r.t the new β).
- If a marked vertex u becomes closed (from being open), then we change β_u to 1 (notice that u remains closed w.r.t the new β).

We call the 4 types of events above as marking, unmarking, opening and closing events.

Now we talk about the order the events happen. When we increase N_u values of ancestors of v continuously, one of the following two events may happen:

- The highest unmarked ancestor u of v may become globally lowest marked, and this may *induce* a closing event for the parent w of u.
- The lowest marked ancestor u of v may become open.

Similarly, when we decrease N_u values of ancestors of v continuously, one of the following two events may happen:

- The lowest marked ancestor u of v may become unmarked (we must that u was lowest marked globally), and this may *induce* an opening event for the parent w of u.
- The lowest marked ancestor u of v may become closed.

Above, if two events happen at the same time, we handle an arbitrary event. Notice that after we handle the event, the conditions for the other event might not hold any more, in which case we do not handle it.

Once we have finished the process of increasing or decreasing N_u values by 1, the clients will be connected to their respective nearest open facilities, breaking ties using the consistent rule. A reconnection happens if a client is connected to a different facility.

Bounding Number of Reconnections. Now we analyze the reconnections made in the algorithm. When a client at $v \in X$ arrives or departs, at most $O(\log D)$ vertices u will have their N_u values changed by 1. We distribute 4 tokens to each ancestor u of v, that are of type-A, type-B, type-C and type-D respectively.[4] We are going to use these tokens to charge the events happened.

First focus on the sequence of marking/unmarking events happened at a vertex u. Right before u becomes unmarked we have $N_u \leq f_u/(2 \times 2^{\mathsf{level}(u)})$ since at the moment we have $\alpha_u = 2$. Immediate after that α_u is changed to 1. For u to become marked again, we need $N_u \leq f_u/2^{\mathsf{level}(u)}$. So during the period N_u must have been increased by at least $f_u/(2 \times 2^{\mathsf{level}(u)})$. Similarly, right before u becomes marked we have $N_u \geq f_u/2^{\mathsf{level}(u)}$ since at the moment we have $\alpha_u = 1$. Then we change α_u to 2 immediately. For u to become unmarked again, N_u should be decreased by at least $f_u/(2 \times 2^{\mathsf{level}(u)})$. So, when a marking/unmarking event happens at u, we can spend $\Omega(f_u/2^{\mathsf{level}(u)})$ type-A tokens owned by u.

Then we focus on the sequence $\mathcal{S}$ of opening/closing events at u between two adjacent marking/unmarking events at u. At these moments, u is marked and $\alpha_u = 2$. For the first event in $\mathcal{S}$, we can spend $\Omega(f_u/2^{\mathsf{level}(u)})$ type-B tokens owned by u. If some opening/closing event e in $\mathcal{S}$ is induced by an unmarking/marking event of some child u' of u, then we can spend $\Omega(f_{u'}/2^{\mathsf{level}(u')}) \geq \Omega(f_u/2^{\mathsf{level}(u)})$ type-C tokens owned by u' for e, and the event e' after e in $\mathcal{S}$ if it exists. Notice that we already argued that u' has collected enough number of type-C tokens.

Then we focus on an event e' in $\mathcal{S}$ such that both e and the event e before e' in $\mathcal{S}$ are not induced. First, assume e is an opening event and e' is a closing event. Then, after e we have $\sum_{u' \in \Lambda_u : u' \text{ unmarked}} N_{u'} = f_u/(2 \times 2^{\mathsf{level}(u)})$ and before e' we have $\sum_{u' \in \Lambda_u : u' \text{ unmarked}} N_{u'} = f_u/(4 \times 2^{\mathsf{level}(u)})$. Notice that the set of unmarked children of u may change, and let U' and U'' be the sets of unmarked children of u at the moments after e and before e' respectively. Again if there is some $u' \in (U' \setminus U'') \cup (U'' \setminus U')$, we spend $\Omega(\frac{f_{u'}}{2^{\mathsf{level}(u')}}) \geq \Omega(\frac{f_u}{2^{\mathsf{level}(u)}})$ type-C tokens owned by u'. Otherwise, $U = U'$ and $f_u/(4 \times 2^{(\mathsf{level}(u))})$ clients in T_u must have departed between e and e' and we can then spend $\Omega(f_u/2^{\mathsf{level}(u)})$ type-D tokens for e'. The case when e is an closing event and e' is an opening event can be argued in the same way.

Thus, whenever an event happens at u, we can spend $\Omega(f_u/2^{\mathsf{level}(u)})$ tokens; moreover if an opening/closing event at u was induced by an unmarking/marking event at some child u' of u, then we can spend $\Omega(f_{u'}/2^{\mathsf{level}(u')})$ tokens for the event at u. A facility u changes its opening status when an event happens at u. Notice that, we reconnect a client only if it was connected to a ready-to-close facility, or it needs to be connected to newly open facility. By Lemma 18, at any moment the number of clients connected to u from outside T_u is at most $O(\log D) \cdot \frac{f_u}{2^{\mathsf{level}(u)}}$. At the moment u changes its opening status because of an non-induced event, then before and after the event the number of clients connected to u from T_u is of order $O\left(\frac{f_u}{2^{\mathsf{level}(u)}}\right)$. u changes its opening status due to a marking/unmarking event happened at some child u' of u, then before and after the event the number of clients connected to u from T_u is of order $\Theta\left(\frac{f_{u'}}{2^{\mathsf{level}(u')}}\right)$. Thus, on average, for each token we spent we connect at most $O(\log D)$ clients. Since each client arrival or departure distributes at most $O(\log D)$ tokens, we have that the amortized number of reconnections (per client arrival/departure) is at most $O(\log^2 D)$.

Analyzing Update Time. Then with the bound on the number of reconnections (recourse), we can bound the update time easily. Indeed, we can maintain a ψ_u for every $u \in V$, which indicates the nearest open facility to u in $T_u \setminus u$ (ψ_u could be undefined). We also maintain

[4] The types are only defined for convenience.

a value N'_u for marked vertices u where $N'_u = \sum_{v \in \Lambda_v, v \text{ unmarked}} N_v$. Whenever a client at v arrives or departs, we need to change $\alpha_u, \beta_u, N_u, N'_u, \psi_u$, marking and opening status of u only for ancestors u of v. The update can be made in $O(\log D)$ time for every client arrival or departure using the information on the vertices. The bottleneck of the algorithm comes from reconnecting clients. We already argued that the amortized number of reconnections per client arrival/departure is $O(\log^2 D)$ and thus it suffices to give an algorithm that can find the clients to be connected efficiently.

For every vertex u, we maintain a double-linked-list of unmarked children u' of u with $N_{u'} \geq 1$. With this structure it is easy to see that for every client that needs to be reconnected, we need $O(\log D)$ time to locate it. If u becomes open, we need to consider each unmarked children u' of u and reconnect clients in $T_{u'}$ to u. The time needed to locate these clients can be made $O(\log D)$ times the number of clients. For every strict ancestor w of u for which there are no open facilities in between we can use the ψ_w information to see if we need to reconnect clients in T_w. If yes, then for every unmarked child w' of w with $N_{w'} \geq 1$ that is not an ancestor of u, we need to connect the clients in $T_{w'}$ to u. Again enumerating these clients takes time $O(\log D)$ times the number of clients. Similarly, if u becomes closed, we then need to connect all clients connected to u to the nearest open facility to u, which can be computed using ψ values of u and its ancestors. Enumerating the clients takes time $O(\log D)$ times the number of clients. Overall, the amortized running time per client arrival/departure is $O(\log^3 D)$.

Nearly Optimal Embeddings of Flat Tori

Ishan Agarwal
Courant Institute of Mathematical Sciences, New York University, NY, USA
ia1020@nyu.edu

Oded Regev
Courant Institute of Mathematical Sciences, New York University, NY, USA

Yi Tang
Courant Institute of Mathematical Sciences, New York University, NY, USA
yt1433@nyu.edu

Abstract

We show that for any n-dimensional lattice $\mathcal{L} \subseteq \mathbb{R}^n$, the torus $\mathbb{R}^n/\mathcal{L}$ can be embedded into Hilbert space with $O(\sqrt{n \log n})$ distortion. This improves the previously best known upper bound of $O(n\sqrt{\log n})$ shown by Haviv and Regev (APPROX 2010, J. Topol. Anal. 2013) and approaches the lower bound of $\Omega(\sqrt{n})$ due to Khot and Naor (FOCS 2005, Math. Ann. 2006).

2012 ACM Subject Classification Mathematics of computing → Discrete mathematics

Keywords and phrases Lattices, metric embeddings, flat torus

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.43

Category APPROX

Funding *Ishan Agarwal*: Research supported by National Science Foundation (NSF) under Grant No. CCF-1814524.
Oded Regev: Research supported by the Simons Collaboration on Algorithms and Geometry, a Simons Investigator Award, and by the National Science Foundation (NSF) under Grant No. CCF-1814524.

1 Introduction

Low distortion embeddings play an important role in many approximation algorithms, allowing one to map points in a "difficult" metric space into another simpler metric space (such as Hilbert space), in a way that approximately preserves distances. See the survey by Indyk [2] for many examples of algorithmic applications. One interesting family of difficult metric spaces is given by *flat tori*. These are defined as quotients of Euclidean space by a lattice, and play an important role in lattice problems and algorithms.

In more detail, an n-dimensional lattice $\mathcal{L} \subseteq \mathbb{R}^n$ is defined as the set of all integer linear combinations of some n linearly independent vectors in $\mathbb{R}^n$. The torus $\mathbb{R}^n/\mathcal{L}$ is the quotient space obtained by identifying points in $\mathbb{R}^n$ with each other if their difference is a lattice vector. The torus has a natural metric associated to it; namely, the distance between any two elements of the torus is defined as the minimum distance between any representative of these elements. So for instance, in the one-dimensional case $\mathbb{R}/\mathbb{Z}$, the distance between 0.1 and 0.9 is 0.2.

Khot and Naor [3] considered the question of how well one can embed flat tori $\mathbb{R}^n/\mathcal{L}$ into Hilbert space. They proved that for any $\mathcal{L}$ and any embedding of $\mathbb{R}^n/\mathcal{L}$ into Hilbert space, the distortion must be at least $\Omega(\frac{\lambda_1(\mathcal{L}^*)}{\mu(\mathcal{L}^*)}\sqrt{n})$. Here, $\mathcal{L}^*$ is the dual lattice of $\mathcal{L}$ and $\lambda_1(\mathcal{L})$ and $\mu(\mathcal{L})$ represent the length of the shortest nonzero vector and the covering radius of $\mathcal{L}$ respectively. It is known by a result of Conway and Thompson (see [4, Page 46]) that, for large enough n, there exist lattices $\mathcal{L}$ where $\lambda_1(\mathcal{L}) = \mu(\mathcal{L})$. Thus the lower bound of Khot and Naor shows that there are n-dimensional lattices whose torus requires distortion


© Ishan Agarwal, Oded Regev, and Yi Tang;
licensed under Creative Commons License CC-BY

$\Omega(\sqrt{n})$ in any embedding into Hilbert space. In the same paper, they also present an embedding that achieves a distortion of $O(n^{3n/2})$ for any lattice $\mathcal{L}$. While the distortion of their embedding might be better than this upper bound, it is known that for some lattices it is super-polynomial [1, Section 7].

In [1] an $O(n\sqrt{\log n})$ distortion metric embedding is constructed, significantly reducing the gap between the upper and lower bounds. They also provide an alternative upper bound of $O(\sqrt{n \log(\mu(\mathcal{L})/\lambda_1(\mathcal{L}))})$. For lattices with good geometric structure (specifically, where the ratio $\mu(\mathcal{L})/\lambda_1(\mathcal{L})$ is only polynomial) this gives an $O(\sqrt{n \log n})$ upper bound. However, in general, the ratio $\mu(\mathcal{L})/\lambda_1(\mathcal{L})$ can be arbitrarily big, in which case this alternative bound is not useful.

Our result is a nearly tight embedding of flat tori, essentially resolving the question of Khot and Naor up to a $\sqrt{\log n}$ factor.

▶ **Theorem 1.1.** *For any lattice $\mathcal{L} \subseteq \mathbb{R}^n$ there exists a metric embedding of $\mathbb{R}^n/\mathcal{L}$ into Hilbert space with distortion $O(\sqrt{n \log n})$.*

1.1 Proof Overview

Our starting point is the embedding by Haviv and Regev [1], which is based on Gaussian measures. Their embedding achieves a distortion of $O(\sqrt{n \log n})$ assuming that the lattice $\mathcal{L}$ has poly(n) "aspect ratio," i.e., the ratio between $\mu(\mathcal{L})$ (the diameter of the torus, or equivalently, the covering radius of the lattice) and $\lambda_1(\mathcal{L})$ (the length of the shortest nonzero vector in the lattice) is polynomial in the dimension n. Their embedding can also be applied to arbitrary lattices; the only issue is that it "saturates" at distance poly$(n)\lambda_1(\mathcal{L})$ – points at greater distance will be contracted by the embedding. See Section 3 for the details.

A natural way to address this issue is to first partition the lattice into scales, and to then embed each scale separately. Specifically, one can define a filtration of sublattices $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$ with each $\mathcal{L}_j$ capturing a different scale of the lattice. Then, for each $j = 1, \ldots, m$, we project the torus on the space orthogonal to $\mathcal{L}_{j-1}$, and embed each projection in Hilbert space separately. Our embedding is then the direct sum of the m individual embeddings.

This approach does work, and is used as part of the construction in [1]. The difficulty is that it introduces an additional $\sqrt{m}$ loss in the distortion, which at worst can be $O(\sqrt{n})$ and is the reason they only achieved an overall distortion of $O(n\sqrt{\log n})$. To see where this loss comes from, consider a short vector inside the span of $\mathcal{L}_1$; this vector only contributes to the first embedding (because it becomes zero in the other $m-1$ projections). On the other hand, a short vector orthogonal to $\mathcal{L}_{m-1}$ gets accounted for in all m projections, leading to an expansion of $\sqrt{m}$ (the square root due to the L_2 norm in the target Hilbert space).

In order to avoid this loss and achieve a $O(\sqrt{n \log n})$ distortion, it is tempting to decompose space into *orthogonal* subspaces (and not nested subspaces as in the above). So instead of projecting on the subspace orthogonal to $\mathcal{L}_{j-1}$, we would like to only project on the subspace of $\mathcal{L}_j$ that is orthogonal to $\mathcal{L}_{j-1}$ (i.e., on the span of $\mathcal{L}_j/\mathcal{L}_{j-1}$). This, however, is impossible; projecting a lattice in such a way in general gives a dense set, and not a lattice.[1]

Our novel contribution is to replace this "harsh" two-sided projection (which is in general impossible) by a more gentle "compressed projection." Namely, we first project orthogonally to $\mathcal{L}_{j-1}$, and then *scale down* the subspace orthogonal to $\mathcal{L}_j$. Returning to the example above,

[1] To see why, consider the two-dimensional lattice generated by $(1, 0)$ and $(\pi, 1)$; its projection on the first coordinate is a dense set.

a short vector orthogonal to $\mathcal{L}_{m-1}$ is still accounted for in all m "compressed projections," but the scaling factors are such that its contributions form a geometric series, so the overall expansion is only a constant instead of $\sqrt{m}$. The technical effort is in showing that these compressions do not distort the geometry by too much; see Section 4 for details. We remark that this "compressed projection" idea might find applications in other cases where decomposing a lattice into scales is desirable.

2 Preliminaries

2.1 Embeddings and Distortion

A metric space is a tuple $(\mathcal{M}, \mathrm{dist}_{\mathcal{M}})$ where $\mathcal{M}$ is a set and $\mathrm{dist}_{\mathcal{M}} : \mathcal{M} \times \mathcal{M} \to \mathbb{R}$ is a function such that the following hold for all $x, y, z \in \mathcal{M}$:

- $\mathrm{dist}_{\mathcal{M}}(x, y) \geq 0$, and the equality holds if and only if $x = y$,
- $\mathrm{dist}_{\mathcal{M}}(x, y) = \mathrm{dist}_{\mathcal{M}}(y, x)$,
- $\mathrm{dist}_{\mathcal{M}}(x, y) + \mathrm{dist}_{\mathcal{M}}(y, z) \geq \mathrm{dist}_{\mathcal{M}}(x, z)$.

For simplicity, we often write metric space $\mathcal{M}$ for $(\mathcal{M}, \mathrm{dist}_{\mathcal{M}})$. We also use dist without the subscript to represent the standard Euclidean metric over $\mathbb{R}^n$ (for some n that is clear from the context). A (metric) embedding is a mapping from one metric space to another.

▶ **Definition 2.1.** *Suppose $F : \mathcal{M}_1 \to \mathcal{M}_2$ is an embedding of metric space $\mathcal{M}_1$ into $\mathcal{M}_2$. The distortion of F is defined by*

$$\inf\left\{ \frac{c_u}{c_l} : \forall x, y \in \mathcal{M}_1,\ c_l \cdot \mathrm{dist}_{\mathcal{M}_1}(x, y) \leq \mathrm{dist}_{\mathcal{M}_2}(F(x), F(y)) \leq c_u \cdot \mathrm{dist}_{\mathcal{M}_1}(x, y) \right\} .$$

2.2 Lattices

We now recall some standard definitions and notations regarding lattices. A (full-rank) *lattice* $\mathcal{L} \subseteq \mathbb{R}^n$ is the set of all integer linear combinations of n linearly independent vectors. This set of vectors is called a *basis* of the lattice. Equivalently, a lattice is a discrete subgroup of the additive group $\mathbb{R}^n$. The *dual lattice* $\mathcal{L}^*$ of $\mathcal{L}$ is defined as the set of all vectors $y \in \mathrm{span}(\mathcal{L})$ such that $\langle x, y \rangle$ is an integer for all vectors $x \in \mathcal{L}$. A *sublattice* $\mathcal{L}' \subseteq \mathcal{L}$ is an additive subgroup of $\mathcal{L}$. We say that a sublattice $\mathcal{L}' \subseteq \mathcal{L}$ is *primitive* if $\mathcal{L}' = \mathcal{L} \cap \mathrm{span}(\mathcal{L}')$. All sublattices in this paper will be primitive. For a lattice $\mathcal{L}$ and a primitive sublattice $\mathcal{L}' \subseteq \mathcal{L}$, the *quotient lattice* $\mathcal{L}/\mathcal{L}'$ is defined as the projection of $\mathcal{L}$ onto the subspace orthogonal to $\mathrm{span}(\mathcal{L}')$. Sublattices and quotient lattices can be thought of as full rank while sitting inside some lower-dimensional space. For lattice $\mathcal{L} \subseteq \mathbb{R}^n$, the *torus* $\mathbb{R}^n/\mathcal{L}$ is naturally associated with the quotient metric, defined as

$$\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \mathrm{dist}(\mathbf{x} - \mathbf{y}, \mathcal{L}) = \min_{\mathbf{v} \in \mathcal{L}} \mathrm{dist}(\mathbf{x} - \mathbf{y}, \mathbf{v}) .$$

The *length of the shortest vector* of a lattice $\mathcal{L}$, denoted by $\lambda_1(\mathcal{L})$, is defined as the minimum length of a non-zero vector in $\mathcal{L}$. Note that here and elsewhere, length refers to the Euclidean norm. The *covering radius* of a lattice $\mathcal{L}$, denoted by $\mu(\mathcal{L})$, is defined as the maximum (Euclidean) distance from any vector in $\mathrm{span}(\mathcal{L})$ to $\mathcal{L}$. Equivalently, as its name suggests, it is the minimum radius such that balls of that radius centered at all lattice points cover the entire $\mathrm{span}(\mathcal{L})$.

We end with two simple technical lemmas, where we denote by π_V the orthogonal projection onto subspace V.

▶ **Lemma 2.2.** *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, vectors $\mathbf{x} \in \mathbb{R}^n$, $\mathbf{v} \in \mathcal{L}$ such that $\|\mathbf{x}-\mathbf{v}\| = \operatorname{dist}(\mathbf{x},\mathcal{L})$, and sublattice $\mathcal{L}' \subseteq \mathcal{L}$,*

$$\|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})\| \leq \mu(\mathcal{L}') \ .$$

Proof. Suppose towards contradiction that $\|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})\| > \mu(\mathcal{L}')$. Then consider the lattice point $\mathbf{u} \in \mathcal{L}'$ that is a closest lattice point to $\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})$ in $\mathcal{L}'$. By definition $\|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})-\mathbf{u}\| \leq \mu(\mathcal{L}')$. Observe that

$$\begin{aligned}\|\mathbf{x}-(\mathbf{v}+\mathbf{u})\|^2 &= \|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v}-\mathbf{u})\|^2 + \|\pi_{\operatorname{span}(\mathcal{L}/\mathcal{L}')}(\mathbf{x}-\mathbf{v}-\mathbf{u})\|^2 \\ &= \|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})-\mathbf{u}\|^2 + \|\pi_{\operatorname{span}(\mathcal{L}/\mathcal{L}')}(\mathbf{x}-\mathbf{v})\|^2 \\ &< \|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})\|^2 + \|\pi_{\operatorname{span}(\mathcal{L}/\mathcal{L}')}(\mathbf{x}-\mathbf{v})\|^2 \\ &= \|\mathbf{x}-\mathbf{v}\|^2 \ ,\end{aligned}$$

which contradicts with the fact that $\|\mathbf{x}-\mathbf{v}\| = \operatorname{dist}(\mathbf{x},\mathcal{L}) = \min_{\mathbf{v}'\in\mathcal{L}} \|\mathbf{x}-\mathbf{v}'\|$. ◀

▶ **Lemma 2.3.** *For any lattice $\mathcal{L}$ and sublattice $\mathcal{L}' \subseteq \mathcal{L}$,*

$$\mu(\mathcal{L})^2 \leq \mu(\mathcal{L}')^2 + \mu(\mathcal{L}/\mathcal{L}')^2 \ .$$

Proof. For any $\mathbf{x} \in \operatorname{span}(\mathcal{L})$, let $\mathbf{v} \in \mathcal{L}$ be a lattice point such that

$$\|\pi_{\operatorname{span}(\mathcal{L}/\mathcal{L}')}(\mathbf{x}-\mathbf{v})\| = \operatorname{dist}(\pi_{\operatorname{span}(\mathcal{L}/\mathcal{L}')}(\mathbf{x}),\mathcal{L}/\mathcal{L}') \ .$$

Without loss of generality it can be assumed that $\|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})\| \leq \mu(\mathcal{L}')$ (since otherwise, we can use $\mathbf{v}+\mathbf{u}$ instead of $\mathbf{v}$, where $\mathbf{u}$ is a closest lattice point to $\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})$ in $\mathcal{L}'$). Then

$$\begin{aligned}\operatorname{dist}(\mathbf{x},\mathcal{L})^2 &\leq \|\mathbf{x}-\mathbf{v}\|^2 \\ &= \|\pi_{\operatorname{span}(\mathcal{L}')}(\mathbf{x}-\mathbf{v})\|^2 + \|\pi_{\operatorname{span}(\mathcal{L}/\mathcal{L}')}(\mathbf{x}-\mathbf{v})\|^2 \\ &\leq \mu(\mathcal{L}')^2 + \mu(\mathcal{L}/\mathcal{L}')^2 \ .\end{aligned}$$

The bound holds for any vector $\mathbf{x}$. Hence $\mu(\mathcal{L})^2 \leq \mu(\mathcal{L}')^2 + \mu(\mathcal{L}/\mathcal{L}')^2$, as desired. ◀

3 Embedding Tori into Hilbert Space

The goal of this section is to prove Lemma 3.6, which summarizes the properties of the Gaussian embedding from [1], including a modified contraction property which we make explicit (see left-hand side of (1)). The proof closely follows that of [1, Theorem 1.4]. We start with some preliminary definitions and results from [1].

For $s > 0$ and $\mathbf{x} \in \mathbb{R}^n$ we define $\rho_s(\mathbf{x}) = \exp(-\pi\|\mathbf{x}/s\|^2)$. For any discrete set A, its Gaussian mass $\rho_s(A)$ is defined as $\sum_{\mathbf{x}\in A} \rho_s(\mathbf{x})$. The *smoothing parameter* of a lattice $\mathcal{L}$ is defined with respect to an $\varepsilon > 0$ and is given by

$$\eta_\varepsilon(\mathcal{L}) = \min\{s : \rho_{1/s}(\mathcal{L}^*) \leq 1+\varepsilon\} \ .$$

▶ **Lemma 3.1** ([1, Lemma 2.5])**.** *For any $n \geq 1$ and lattice $\mathcal{L} \subseteq \mathbb{R}^n$, $\eta_\varepsilon(\mathcal{L}^*) \leq \frac{2\sqrt{n}}{\lambda_1(\mathcal{L})}$ where $\varepsilon = 2^{-10n}$.*

Consider the function

$$h_{\mathcal{L},s}(\mathbf{x}) = 1 - \frac{\rho_s(\mathcal{L}-\mathbf{x})}{\rho_s(\mathcal{L})} \ .$$

Below we list some basic properties of this function, which ideally we would like to be proportional to the squared distance from the lattice. This is indeed the case, assuming the distance is not too large compared to s, and that s itself is small compared to the geometry of the lattice. The upper bound is shown in Item 1, and the lower bound is established in Items 2 and 3 (which give very similar bounds). When the distance is sufficiently larger than s, the function reaches saturation, as shown in Item 4.

▶ **Lemma 3.2** ([1, Lemmas 3.1 and 3.2]). *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, $s > 0$, and vector $\mathbf{x} \in \mathbb{R}^n$,*

1. $s^2 \cdot h_{\mathcal{L},s}(\mathbf{x}) \leq \pi \cdot \mathrm{dist}(\mathbf{x}, \mathcal{L})^2$,
2. $s^2 \cdot h_{\mathcal{L},s}(\mathbf{x}) \geq c \cdot \mathrm{dist}(\mathbf{x}, \mathcal{L})^2$ *if* $s \leq \frac{1}{2\,\eta_\varepsilon(\mathcal{L}^*)}$ *for some* $0 < \varepsilon \leq \frac{1}{1000}$ *and* $\mathrm{dist}(\mathbf{x}, \mathcal{L}) \leq \frac{s}{\sqrt{2}}$, *where c is an absolute constant,*
3. $h_{\mathcal{L},s}(\mathbf{x}) \geq 1 - e^{-\pi\,\mathrm{dist}(\mathbf{x},\mathcal{L})^2/s^2} - 2^{-11n}$ *if* $\lambda_1(\mathcal{L}) \geq 4\sqrt{n} \cdot s$,
4. $h_{\mathcal{L},s}(\mathbf{x}) \geq 1 - 2^{-11n}$ *if* $\mathrm{dist}(\mathbf{x}, \mathcal{L}) > 2\sqrt{n} \cdot s$.

▶ **Definition 3.3** ([1, Section 5]). *For lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and $s > 0$, the embedding $H_{\mathcal{L},s} : \mathbb{R}^n/\mathcal{L} \to L_2(\mathbb{R}^n/\mathcal{L})$ maps vector $\mathbf{x} \in \mathbb{R}^n$ to the function $H_{\mathcal{L},s}(\mathbf{x}) \in L_2(\mathbb{R}^n/\mathcal{L})$ given by*

$$H_{\mathcal{L},s}(\mathbf{x})(\mathbf{y}) = \frac{s}{\sqrt{2\rho_s(\mathcal{L})}} \left(\frac{2}{s}\right)^{n/2} \rho_{\frac{s}{\sqrt{2}}}(\mathcal{L} + \mathbf{y} - \mathbf{x}) \ .$$

▶ **Lemma 3.4** ([1, Proposition 5.1]). *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, $s > 0$, and vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, $\mathrm{dist}_{L_2(\mathbb{R}^n/\mathcal{L})}(H_{\mathcal{L},s}(\mathbf{x}), H_{\mathcal{L},s}(\mathbf{y}))^2 = s^2 \cdot h_{\mathcal{L},s}(\mathbf{x} - \mathbf{y})$.*

▶ **Definition 3.5** ([1, Section 5.1]). *For lattice $\mathcal{L} \subseteq \mathbb{R}^n$, $s > 0$, and $k \geq 1$, the embedding $H^{(k)}_{\mathcal{L},s}$ is defined by $H^{(k)}_{\mathcal{L},s} = (H_{\mathcal{L},s_1}, \ldots, H_{\mathcal{L},s_k})$ where $s_i = 2^{i-1}s$. We often take $s = \lambda_1(\mathcal{L})/(4\sqrt{n})$, in which case we omit the subscript s and simply write $H^{(k)}_{\mathcal{L}}$.*

▶ **Lemma 3.6.** *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, $k \geq 1$, and vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,*

$$\begin{aligned} \frac{c_H}{n} \cdot \min(\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x}, \mathbf{y}), 2^{k-1}\,\lambda_1(\mathcal{L}))^2 &\leq \mathrm{dist}_{L_2(\mathbb{R}^n/\mathcal{L})^k}(H^{(k)}_{\mathcal{L}}(\mathbf{x}), H^{(k)}_{\mathcal{L}}(\mathbf{y}))^2 \\ &\leq \pi k \cdot \mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x}, \mathbf{y})^2 \ , \end{aligned} \tag{1}$$

where $c_H > 0$ is an absolute constant.

Proof. By Lemma 3.4, and recalling the notation $s_i = 2^{i-1}s$ where $s = \frac{\lambda_1(\mathcal{L})}{4\sqrt{n}}$,

$$\begin{aligned} \mathrm{dist}_{L_2(\mathbb{R}^n/\mathcal{L})^k}(H^{(k)}_{\mathcal{L}}(\mathbf{x}), H^{(k)}_{\mathcal{L}}(\mathbf{y}))^2 &= \sum_{i=1}^{k} \mathrm{dist}_{L_2(\mathbb{R}^n/\mathcal{L})}(H_{\mathcal{L},s_i}(\mathbf{x}), H_{\mathcal{L},s_i}(\mathbf{y}))^2 \\ &= \sum_{i=1}^{k} s_i^2 \cdot h_{\mathcal{L},s_i}(\mathbf{x} - \mathbf{y}) \ . \end{aligned}$$

Noting that $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x}, \mathbf{y}) = \mathrm{dist}(\mathbf{x} - \mathbf{y}, \mathcal{L})$, the upper bound in (1) follows from Item 1 in Lemma 3.2:

$$\sum_{i=1}^{k} s_i^2 \cdot h_{\mathcal{L},s_i}(\mathbf{x} - \mathbf{y}) \leq \sum_{i=1}^{k} \pi \cdot \mathrm{dist}(\mathbf{x} - \mathbf{y}, \mathcal{L})^2 = \pi k \cdot \mathrm{dist}(\mathbf{x} - \mathbf{y}, \mathcal{L})^2 \ .$$

For the lower bound in (1), we will show that for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$, there exists $i \in \{1, \ldots, k\}$ such that

$$s_i^2 \cdot h_{\mathcal{L},s_i}(\mathbf{x} - \mathbf{y}) \geq \frac{c_H}{n} \cdot \min(\mathrm{dist}(\mathbf{x} - \mathbf{y}, \mathcal{L}), 2^{k-1}\,\lambda_1(\mathcal{L}))^2 \ . \tag{2}$$

We consider three cases.

1. $\operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L}) \le \frac{\lambda_1(\mathcal{L})}{4\sqrt{2n}} = \frac{s}{\sqrt{2}}$. Note that according to Lemma 3.1, $s \le \frac{1}{2\,\eta_\varepsilon(\mathcal{L}^*)}$ for some $0 < \varepsilon \le \frac{1}{1000}$. Then by Item 2 of Lemma 3.2,

$$s^2 \cdot h_{\mathcal{L},s}(\mathbf{x}-\mathbf{y}) \ge c \cdot \operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L})^2 \ge \frac{c}{n} \cdot \operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L})^2 ,$$

   which proves (2) with $i = 1$.
2. $\frac{s}{\sqrt{2}} < \operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L}) \le \lambda_1(\mathcal{L}) = 4\sqrt{n} \cdot s$. By Item 3 of Lemma 3.2,

$$\begin{aligned} s^2 \cdot h_{\mathcal{L},s}(\mathbf{x}-\mathbf{y}) &\ge s^2 \cdot (1 - e^{-\pi/2} - 2^{-11n}) \\ &= \frac{1 - e^{-\pi/2} - 2^{-11n}}{16n} \cdot \lambda_1(\mathcal{L})^2 \\ &\ge \frac{1 - e^{-\pi/2} - 2^{-11n}}{16n} \cdot \operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L})^2 , \end{aligned}$$

   which again proves (2) with $i = 1$.
3. $\operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L}) > 4\sqrt{n} \cdot s$. Let $j \in \{2,\dots,k\}$ be the largest index such that $2\sqrt{n} \cdot s_j < \operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L})$. Notice that if $j < k$ then $4\sqrt{n} \cdot s_j = 2\sqrt{n} \cdot s_{j+1} \ge \operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L})$, and that if $j = k$, then $4\sqrt{n} \cdot s_j = 2^{k-1}\,\lambda_1(\mathcal{L})$. Then by Item 4 of Lemma 3.2,

$$\begin{aligned} s_j^2 \cdot h_{\mathcal{L},s_j}(\mathbf{x}-\mathbf{y}) &\ge s_j^2 \cdot (1 - 2^{-11n}) \\ &= \frac{1 - 2^{-11n}}{16n} \cdot (4\sqrt{n} \cdot s_j)^2 \\ &\ge \frac{1 - 2^{-11n}}{16n} \cdot \min(\operatorname{dist}(\mathbf{x}-\mathbf{y},\mathcal{L}), 2^{k-1}\,\lambda_1(\mathcal{L}))^2 , \end{aligned}$$

   which proves (2) with $i = j$. ◀

4 Embedding into Tori

The goal of this section is to prove Lemma 4.13, which shows that there exists an embedding from an arbitrary torus into a tuple of tori with good geometry. The embedding is constructed based on "good filtrations," which we define and instantiate in Section 4.1. The definition of the embedding is given in Section 4.2, and its expansion and contraction properties are shown in Section 4.3 and Section 4.4 respectively. The contraction property matches the modified notion of contraction used in Lemma 3.6.

4.1 Good Filtrations

In this section we define the notion of a (q,γ)-*filtration* (Definition 4.1) and show how to construct a good one for every lattice (Lemma 4.3). We also include a small technical lemma that will be useful later (Lemma 4.4).

A *filtration* of a lattice $\mathcal{L}$ is a chain of sublattices $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$. We call m the *size* of the filtration.

▶ **Definition 4.1.** *For $q \ge 1$, $\gamma > 1$, we say that a filtration $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$ is a* (q,γ)-filtration *if it satisfies both*

1. $\mu(\mathcal{L}_j/\mathcal{L}_{j-1}) \le q\,\lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})/2$ *for all* $1 \le j \le m$, *and*
2. $\lambda_1(\mathcal{L}_{j+1}/\mathcal{L}_j) \ge \gamma\,\lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})$ *for all* $1 \le j < m$.

Our construction of good filtrations is based on Korkine-Zolotarev bases, defined next. Recall first that for a sequence of vectors $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, its *Gram-Schmidt orthogonalization* $(\mathbf{b}'_1, \ldots, \mathbf{b}'_n)$ is defined by

$$\mathbf{b}'_i = \mathbf{b}_i - \sum_{j=1}^{i-1} \mu_{i,j} \mathbf{b}'_j \ , \quad \text{where } \mu_{i,j} = \frac{\langle \mathbf{b}_i, \mathbf{b}'_j \rangle}{\langle \mathbf{b}'_j, \mathbf{b}'_j \rangle} \ ,$$

i.e., $\mathbf{b}'_i$ is the projection of $\mathbf{b}_i$ on the space orthogonal to $\text{span}(\mathbf{b}_1, \ldots, \mathbf{b}_{i-1})$.

▶ **Definition 4.2.** *A basis $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ for a lattice $\mathcal{L} \subseteq \mathbb{R}^n$ is called a* Korkine-Zolotarev basis *if*

- *$\mathbf{b}'_i$ is a shortest vector of $\mathcal{L}/\mathcal{L}_{i-1}$ for all $1 \le i \le n$, and*
- *$|\mu_{i,j}| \le 1/2$ for all $1 \le j < i \le n$,*

where $(\mathbf{b}'_1, \ldots, \mathbf{b}'_n)$ is the Gram-Schmidt orthogonalization of $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$, $\mu_{i,j}$ are the corresponding coefficients, and $\mathcal{L}_i$ is the lattice generated by $(\mathbf{b}_1, \ldots, \mathbf{b}_i)$ (with $\mathcal{L}_0 = \{\mathbf{0}\}$).

It is easy to prove that a Korkine-Zolotarev basis exists for any lattice. We remark that the second property above will not be used in this paper.

▶ **Lemma 4.3.** *For any $n \ge 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, and $\gamma > 1$, there exists a $(\gamma\sqrt{n}, \gamma)$-filtration of $\mathcal{L}$.*

Proof. Let $(\mathbf{b}_1, \ldots, \mathbf{b}_n)$ be a Korkine-Zolotarev basis of $\mathcal{L}$. Let $(\mathbf{b}'_1, \ldots, \mathbf{b}'_n)$ be its Gram-Schmidt orthogonalization, and consider the filtration $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_n = \mathcal{L}$ where $\mathcal{L}_i$ is the lattice generated by $(\mathbf{b}_1, \ldots, \mathbf{b}_i)$. From Definition 4.2 we know that $\lambda_1(\mathcal{L}/\mathcal{L}_{i-1}) = \|\mathbf{b}'_i\| = \lambda_1(\mathcal{L}_k/\mathcal{L}_{i-1})$, for all $1 \le i \le k \le n$. Construct a coarsening of this filtration, $\{\mathbf{0}\} = \mathcal{L}_{i_0} \subset \mathcal{L}_{i_1} \subset \cdots \subset \mathcal{L}_{i_m} = \mathcal{L}$, as follows. Let $i_0 = 0$ and for $j \ge 1$, $i_j \in \{i_{j-1}+1, \ldots, n\}$ be the largest index such that $\|\mathbf{b}'_k\| \le \gamma \|\mathbf{b}'_{i_{j-1}+1}\|$ for all $k \in \{i_{j-1}+1, \ldots, i_j\}$. Finally, stop when $i_m = n$. We are going to show that this coarser filtration is a $(\gamma\sqrt{n}, \gamma)$-filtration.

We observe that for all $1 \le j \le m$,

$$\lambda_1(\mathcal{L}_{i_j}/\mathcal{L}_{i_{j-1}}) = \|\mathbf{b}'_{i_{j-1}+1}\| \ .$$

Then, by construction of the coarsening, for all $1 \le j < m$,

$$\lambda_1(\mathcal{L}_{i_{j+1}}/\mathcal{L}_{i_j}) = \|\mathbf{b}'_{i_j+1}\| > \gamma \|\mathbf{b}'_{i_{j-1}+1}\| = \gamma \, \lambda_1(\mathcal{L}_{i_j}/\mathcal{L}_{i_{j-1}}) \ .$$

This proves the second property of a $(\gamma\sqrt{n}, \gamma)$-filtration. Moreover,

$$\begin{aligned}
\mu(\mathcal{L}_{i_j}/\mathcal{L}_{i_{j-1}})^2 &\le \sum_{k=i_{j-1}+1}^{i_j} \mu(\mathcal{L}_k/\mathcal{L}_{k-1})^2 \\
&= \sum_{k=i_{j-1}+1}^{i_j} \|\mathbf{b}'_k\|^2/4 \\
&\le \sum_{k=i_{j-1}+1}^{i_j} \gamma^2 \|\mathbf{b}'_{i_{j-1}+1}\|^2/4 \\
&\le \gamma^2 n \cdot \lambda_1(\mathcal{L}_{i_j}/\mathcal{L}_{i_{j-1}})^2/4 \ ,
\end{aligned}$$

where the first inequality is by Lemma 2.3 and the second inequality is by construction of the coarsening. This proves the first property of a $(\gamma\sqrt{n}, \gamma)$-filtration. ◀

We end by proving a small property of (q, γ)-filtrations.

▶ **Lemma 4.4.** *For any (q,γ)-filtration $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$ and $1 \le j \le m$,*

$$\mu(\mathcal{L}_j) \le \frac{q}{\sqrt{1-1/\gamma^2}} \cdot \lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})/2 \ .$$

Consequently, if $\gamma \ge 2$, then $\mu(\mathcal{L}_j) \le q\,\lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})$.

Proof. The inequality can be proved as follows:

$$\begin{aligned}
\mu^2(\mathcal{L}_j) &\le \sum_{i=1}^{j} \mu^2(\mathcal{L}_i/\mathcal{L}_{i-1}) \\
&\le \sum_{i=1}^{j} q^2\,\lambda_1^2(\mathcal{L}_i/\mathcal{L}_{i-1})/4 \\
&\le \sum_{i=1}^{j} \frac{q^2}{\gamma^{2(j-i)}} \cdot \lambda_1^2(\mathcal{L}_j/\mathcal{L}_{j-1})/4 \\
&\le \frac{q^2}{1-1/\gamma^2} \cdot \lambda_1^2(\mathcal{L}_j/\mathcal{L}_{j-1})/4 \ ,
\end{aligned}$$

where the first inequality uses Lemma 2.3, the second inequality uses the first property in Definition 4.1, and the third inequality uses the second property in Definition 4.1. ◀

4.2 The Embedding

Let $\mathcal{F}$ be a filtration $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$ of a lattice $\mathcal{L} \subseteq \mathbb{R}^n$. The filtration naturally induces an orthogonal decomposition of $\mathbb{R}^n$ into m subspaces, namely, $\mathrm{span}(\mathcal{L}_j/\mathcal{L}_{j-1})$ for $j = 1, \ldots, m$. We use $\pi_{\mathcal{F},j}$ to denote $\pi_{\mathrm{span}(\mathcal{L}_j/\mathcal{L}_{j-1})}$, the projection on the j-th subspace. We will similarly use $\pi^{\ge}_{\mathcal{F},j}$, $\pi^{<}_{\mathcal{F},j}$, $\pi^{>}_{\mathcal{F},j}$, and $\pi^{\le}_{\mathcal{F},j}$ to denote projections on the span of prefixes and suffixes of this decomposition. Specifically, for $1 \le j \le m$ we have $\pi^{\ge}_{\mathcal{F},j} = \pi_{\mathrm{span}(\mathcal{L}/\mathcal{L}_{j-1})}$, $\pi^{<}_{\mathcal{F},j} = \pi_{\mathrm{span}(\mathcal{L}_{j-1})}$, $\pi^{>}_{\mathcal{F},j} = \pi_{\mathrm{span}(\mathcal{L}/\mathcal{L}_j)}$, and $\pi^{\le}_{\mathcal{F},j} = \pi_{\mathrm{span}(\mathcal{L}_j)}$.

▶ **Definition 4.5.** *For filtration $\mathcal{F}$ of size m, $0 < \alpha < 1$, and $1 \le j \le m$, the embedding $E_{\mathcal{F},\alpha,j}$ is defined by*

$$E_{\mathcal{F},\alpha,j}(\mathbf{x}) = \sum_{i=j}^{m} \alpha^{i-j} \pi_{\mathcal{F},i}(\mathbf{x}) \ .$$

Note that since $E_{\mathcal{F},\alpha,j}$ is linear, for any lattice $\mathcal{L} \subseteq \mathbb{R}^n$ and vector $\mathbf{x} \in \mathbb{R}^n$, $E_{\mathcal{F},\alpha,j}(\mathbf{x}+\mathcal{L}) = E_{\mathcal{F},\alpha,j}(\mathbf{x}) + E_{\mathcal{F},\alpha,j}(\mathcal{L})$, and thus $E_{\mathcal{F},\alpha,j}$ is a well-defined embedding from the torus $\mathbb{R}^n/\mathcal{L}$ to the torus $E_{\mathcal{F},\alpha,j}(\mathbb{R}^n/\mathcal{L})$.

▶ **Definition 4.6.** *For a filtration $\mathcal{F}$ of size m and $0 < \alpha < 1$, the embedding $E_{\mathcal{F},\alpha}$ is defined by $E_{\mathcal{F},\alpha} = (E_{\mathcal{F},\alpha,1}, \ldots, E_{\mathcal{F},\alpha,m})$ with the metric being ℓ_2 of the tori metrics.*

4.3 Expansion of the Embedding

▶ **Definition 4.7** (Realization of distance in torus)**.** *For any $n \ge 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, and vector $\mathbf{x} \in \mathbb{R}^n$, since $\mathrm{dist}(\mathbf{x},\mathcal{L}) = \min_{\mathbf{v}\in\mathcal{L}} \|\mathbf{x}-\mathbf{v}\|$, there always exists $\mathbf{v} \in \mathcal{L}$ such that $\mathrm{dist}(\mathbf{x},\mathcal{L}) = \|\mathbf{x}-\mathbf{v}\|$. We say such minimizer $\mathbf{v}$* realizes *the distance $\mathrm{dist}(\mathbf{x},\mathcal{L})$. Similarly, for vectors $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$, we say $\mathbf{v}$* realizes *the distance $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})$ if $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}-\mathbf{v}\|$.*

▶ **Lemma 4.8** (Expansion of the embedding). *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$ with filtration $\mathcal{F}$, $0 < \alpha < 1$, and vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$,*

$$\begin{aligned}\mathrm{dist}_{E_{\mathcal{F},\alpha}(\mathbb{R}^n/\mathcal{L})}(E_{\mathcal{F},\alpha}(\mathbf{x}), E_{\mathcal{F},\alpha}(\mathbf{y}))^2 &:= \sum_{j=1}^{m} \mathrm{dist}_{E_{\mathcal{F},\alpha,j}(\mathbb{R}^n/\mathcal{L})}(E_{\mathcal{F},\alpha,j}(\mathbf{x}), E_{\mathcal{F},\alpha,j}(\mathbf{y}))^2 \\ &\leq \frac{1}{1-\alpha^2} \cdot \mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})^2 \ .\end{aligned}$$

Proof. Let m be the size of $\mathcal{F}$. For all $\mathbf{v} \in \mathcal{L}$, the embedded distance can be bounded from above by

$$\begin{aligned}\sum_{j=1}^{m} \mathrm{dist}_{E_{\mathcal{F},\alpha,j}(\mathbb{R}^n/\mathcal{L})}(E_{\mathcal{F},\alpha,j}(\mathbf{x}), E_{\mathcal{F},\alpha,j}(\mathbf{y}))^2 &\leq \sum_{j=1}^{m} \|E_{\mathcal{F},\alpha,j}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 \\ &= \sum_{j=1}^{m}\sum_{i=j}^{m} \alpha^{2(i-j)} \|\pi_{\mathcal{F},i}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 \\ &\leq \frac{1}{1-\alpha^2} \cdot \sum_{i=1}^{m} \|\pi_{\mathcal{F},i}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 \\ &= \frac{1}{1-\alpha^2} \cdot \|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2 \ ,\end{aligned}$$

which, for $\mathbf{v}$ realizing $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})$, gives $\frac{1}{1-\alpha^2} \cdot \mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})^2$ as desired. ◀

4.4 Contraction of the Embedding

▶ **Lemma 4.9.** *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$, lattice point $\mathbf{v}' \in \mathcal{L}$ realizing $\mathrm{dist}(\mathbf{x},\mathcal{L})$, and lattice point $\mathbf{v} \in \mathcal{L}$,*

$$\|\mathbf{x}-\mathbf{v}\| \geq \frac{1}{2}\|\mathbf{v}-\mathbf{v}'\| \ .$$

Consequently, if $\mathbf{v}$ does not realize $\mathrm{dist}(\mathbf{x},\mathcal{L})$, then $\mathbf{v} \neq \mathbf{v}'$ and

$$\|\mathbf{x}-\mathbf{v}\| \geq \frac{1}{2}\,\lambda_1(\mathcal{L}) \ .$$

Proof. By definition, $\|\mathbf{x}-\mathbf{v}'\| \leq \|\mathbf{x}-\mathbf{v}\|$. Then by the triangle inequality, $\|\mathbf{v}-\mathbf{v}'\| \leq \|\mathbf{x}-\mathbf{v}\| + \|\mathbf{x}-\mathbf{v}'\| \leq 2\|\mathbf{x}-\mathbf{v}\|$, as desired. ◀

▶ **Lemma 4.10.** *For any (q,γ)-filtration $\mathcal{F}$ given by $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$, $1/\gamma \leq \alpha < 1$, and $1 \leq j \leq m$, $\lambda_1(E_{\mathcal{F},\alpha,j}(\mathcal{L})) = \lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})$.*

Proof. We prove the claim by induction on j. When $j = m$, $E_{\mathcal{F},\alpha,m}(\mathcal{L}) = \mathcal{L}_m/\mathcal{L}_{m-1}$, and thus the claim holds trivially.

Suppose the claim holds for $j+1$. Then for j, note that $\mathcal{L}_j/\mathcal{L}_{j-1} \subseteq E_{\mathcal{F},\alpha,j}(\mathcal{L})$. Therefore $\lambda_1(E_{\mathcal{F},\alpha,j}(\mathcal{L}))$ is the minimum of $\lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})$ and the minimum length of vectors in the set $E_{\mathcal{F},\alpha,j}(\mathcal{L}) \setminus (\mathcal{L}_j/\mathcal{L}_{j-1})$. Since $E_{\mathcal{F},\alpha,j} = \pi_{\mathcal{F},j} + \alpha E_{\mathcal{F},\alpha,j+1}$, the length of any vector in $E_{\mathcal{F},\alpha,j}(\mathcal{L}) \setminus (\mathcal{L}_j/\mathcal{L}_{j-1})$ is bounded from below by

$$\begin{aligned}\alpha\,\lambda_1(E_{\mathcal{F},\alpha,j+1}(\mathcal{L})) &= \alpha\,\lambda_1(\mathcal{L}_{j+1}/\mathcal{L}_j) \\ &\geq \alpha\gamma\,\lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1}) \\ &\geq \lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1}) \ ,\end{aligned}$$

where the equality is the induction assumption and the first inequality uses the second property in Definition 4.1. Hence $\lambda_1(E_{\mathcal{F},\alpha,j}(\mathcal{L})) = \lambda_1(\mathcal{L}_j/\mathcal{L}_{j-1})$, as desired. ◀

Combining Lemma 4.10 with Lemma 4.4 as well as Definition 4.1, we immediately get the following corollary.

▶ **Corollary 4.11.** *For any (q,γ)-filtration $\mathcal{F}$ given by $\{\mathbf{0}\} = \mathcal{L}_0 \subset \mathcal{L}_1 \subset \cdots \subset \mathcal{L}_m = \mathcal{L}$ with $\gamma \geq 2$ and $1/\gamma \leq \alpha < 1$,*
1. *$\mu(\mathcal{L}_j) \leq q\,\lambda_1(E_{\mathcal{F},\alpha,j}(\mathcal{L}))$ for all $1 \leq j \leq m$, and*
2. *$\lambda_1(E_{\mathcal{F},\alpha,j+1}(\mathcal{L})) \geq \gamma\,\lambda_1(E_{\mathcal{F},\alpha,j}(\mathcal{L}))$ for all $1 \leq j < m$.*

▶ **Lemma 4.12** (Contraction of the embedding)**.** *For any $n \geq 1$, lattice $\mathcal{L} \subseteq \mathbb{R}^n$ with (q,γ)-filtration $\mathcal{F}$ of size m satisfying $\gamma \geq 2$ and $q \leq \gamma^2/32$, $\frac{1}{2} \leq \alpha < 1$, and vectors $\mathbf{x},\mathbf{y} \in \mathbb{R}^n$,*

$$\sum_{j=1}^{m} \min\Big(\mathrm{dist}_{E_{\mathcal{F},\alpha,j}(\mathbb{R}^n/\mathcal{L})}(E_{\mathcal{F},\alpha,j}(\mathbf{x}), E_{\mathcal{F},\alpha,j}(\mathbf{y})), q^2\,\lambda_1(E_{\mathcal{F},\alpha,j}(\mathcal{L}))\Big)^2 \geq c_E \cdot \mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})^2\,, \quad (3)$$

where $c_E > 0$ is an absolute constant.

Proof. For simplicity, we omit the subscript $\mathcal{F}$ in the notations $\pi_{\mathcal{F},j}$, $\pi^{\geqslant}_{\mathcal{F},j}$, $\pi^{<}_{\mathcal{F},j}$, $\pi^{>}_{\mathcal{F},j}$, $\pi^{\leqslant}_{\mathcal{F},j}$, $E_{\mathcal{F},\alpha,j}$ and $E_{\mathcal{F},\alpha}$ in this proof.

Let $\mathbf{v} \in \mathcal{L}$ be a lattice point that realizes $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})$. Then $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}-\mathbf{v}\|$. Hence our goal is equivalently to show that the left-hand side of (3) satisfies

$$\sum_{j=1}^{m} \min\Big(\mathrm{dist}_{E_{\alpha,j}(\mathbb{R}^n/\mathcal{L})}(E_{\alpha,j}(\mathbf{x}), E_{\alpha,j}(\mathbf{y})), q^2\,\lambda_1(E_{\alpha,j}(\mathcal{L}))\Big)^2 \geq c_E \cdot \|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2\,. \quad (4)$$

Let $j_1 \in \{0,1,\ldots,m\}$ be the smallest index satisfying that for all $j \in \{j_1+1,\ldots,m\}$, $E_{\alpha,j}(\mathbf{v})$ realizes $\mathrm{dist}_{E_{\alpha,j}(\mathbb{R}^n/\mathcal{L})}(E_{\alpha,j}(\mathbf{x}), E_{\alpha,j}(\mathbf{y}))$. Then for all $j \in \{j_1+1,\ldots,m\}$,

$$\begin{aligned}\mathrm{dist}_{E_{\alpha,j}(\mathbb{R}^n/\mathcal{L})}(E_{\alpha,j}(\mathbf{x}), E_{\alpha,j}(\mathbf{y})) &= \|E_{\alpha,j}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \\ &\geq \|\pi_j(\mathbf{x}-\mathbf{y}-\mathbf{v})\|\,.\end{aligned} \quad (5)$$

Moreover, according to Lemma 2.2 and Corollary 4.11,

$$\|\pi_j(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \leq \|\pi^{\leqslant}_j(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \leq \mu(\mathcal{L}_j) \leq q\,\lambda_1(E_{\alpha,j}(\mathcal{L})) \leq q^2\,\lambda_1(E_{\alpha,j}(\mathcal{L}))\,. \quad (6)$$

Combining (5) and (6), the left-hand side of (4) is bounded from below by

$$\sum_{j=j_1+1}^{m} \|\pi_j(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 = \|\pi^{>}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2\,. \quad (7)$$

If it is the case that

$$\|\pi^{>}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 \geq \frac{1}{2}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2\,,$$

then (7) clearly suffices to prove (4). So from now on we assume that

$$\begin{aligned}&\|\pi^{>}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 < \frac{1}{2}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2\,,\\ \text{i.e., }&\|\pi^{\leqslant}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 > \frac{1}{2}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2\,.\end{aligned} \quad (8)$$

In particular, this implies $j_1 > 0$. Then, by definition of j_1, $E_{\alpha,j_1}(\mathbf{v})$ does not realize $\mathrm{dist}_{E_{\alpha,j_1}(\mathbb{R}^n/\mathcal{L})}(E_{\alpha,j_1}(\mathbf{x}), E_{\alpha,j_1}(\mathbf{y}))$, which, by using Lemma 4.9 with lattice $E_{\alpha,j_1}(\mathcal{L})$, implies

$$\|E_{\alpha,j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \geq \frac{1}{2}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L}))\,. \quad (9)$$

Under assumption (8), it suffices to prove that there exists an index $j_0 \in \{1, \ldots, m\}$ such that

$$\min\left(\mathrm{dist}_{E_{\alpha,j_0}(\mathbb{R}^n/\mathcal{L})}(E_{\alpha,j_0}(\mathbf{x}), E_{\alpha,j_0}(\mathbf{y})), q^2\,\lambda_1(E_{\alpha,j_0}(\mathcal{L}))\right)^2 \geq c \cdot \|\pi^{\leq}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 ,$$

or equivalently,

$$\|E_{\alpha,j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|^2 \geq c \cdot \|\pi^{\leq}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 \text{ , and} \tag{10}$$

$$q^4\,\lambda_1^2(E_{\alpha,j_0}(\mathcal{L})) \geq c \cdot \|\pi^{\leq}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 , \tag{11}$$

where $\mathbf{v}' \in \mathcal{L}$ is a lattice point such that $E_{\alpha,j_0}(\mathbf{v}')$ realizes $\mathrm{dist}_{E_{\alpha,j_0}(\mathbb{R}^n/\mathcal{L})}(E_{\alpha,j_0}(\mathbf{x}), E_{\alpha,j_0}(\mathbf{y}))$, and c is some absolute constant. Note that, without loss of generality, it can be assumed that

$$\|\pi^{<}_{j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v}')\| \leq \mu(\mathcal{L}_{j_0-1}) \tag{12}$$

(since otherwise, we can use $\mathbf{v}'+\mathbf{u}$ instead of $\mathbf{v}'$, where $\mathbf{u} \in \mathcal{L}_{j_0-1}$ realizes $\mathrm{dist}(\pi^{<}_{j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v}'), \mathcal{L}_{j_0-1})$).

We choose $j_0 = j_1$ if

$$\mu(\mathcal{L}_{j_1-1}) \leq \frac{1}{4}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L})) ,$$

and otherwise $j_0 = j_1 - 1$. By Corollary 4.11 and the condition $q \leq \gamma^2/32$, we know that

$$\mu(\mathcal{L}_{j_1-2}) \leq \frac{q}{\gamma^2} \cdot \lambda_1(E_{\alpha,j_1}(\mathcal{L})) \leq \frac{1}{32}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L})) .$$

Moreover, as $\mathcal{L}_{j_1}/\mathcal{L}_{j_1-1}$ is both a quotient of $\mathcal{L}_{j_1}$ and a sublattice of $E_{\alpha,j_1}(\mathcal{L})$,

$$\mu(\mathcal{L}_{j_1}) \geq \mu(\mathcal{L}_{j_1}/\mathcal{L}_{j_1-1}) \geq \frac{1}{2}\,\lambda_1(\mathcal{L}_{j_1}/\mathcal{L}_{j_1-1}) \geq \frac{1}{2}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L}))$$

(and the last inequality is actually an equality due to Lemma 4.10). Therefore j_0 satisfies

$$\mu(\mathcal{L}_{j_0-1}) \leq \frac{1}{4}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L})) \text{ , and} \tag{13}$$

$$\mu(\mathcal{L}_{j_0}) > \frac{1}{4}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L})) . \tag{14}$$

We first prove (11) for this choice of j_0:

$$\begin{aligned} q^2\,\lambda_1(E_{\alpha,j_0}(\mathcal{L})) &\geq q\,\mu(\mathcal{L}_{j_0}) \\ &> \frac{q}{4} \cdot \lambda_1(E_{\alpha,j_1}(\mathcal{L})) \\ &\geq \frac{1}{4}\,\mu(\mathcal{L}_{j_1}) \\ &\geq \frac{1}{4}\|\pi^{\leq}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| , \end{aligned}$$

where the first and third inequalities use Corollary 4.11, the second inequality follows from (14), and the last inequality uses Lemma 2.2.

We next prove (10) for this choice of j_0. We begin with showing that

$$\pi^{\geq}_{j_2}(\mathbf{v}') = \pi^{\geq}_{j_2}(\mathbf{v}) , \tag{15}$$

where $j_2 = \min(j_1+1, m)$. Suppose towards contradiction that $\pi^{>}_{j_2}(\mathbf{v}') \neq \pi^{>}_{j_2}(\mathbf{v})$ (implying $j_2 < m$, and thus $j_2 = j_1+1$). Then, by definition, $\|E_{\alpha,j_1+2}(\mathbf{v}-\mathbf{v}')\| \geq \lambda_1(E_{\alpha,j_1+2}(\mathcal{L}))$. Hence

$$\begin{aligned}
\|\pi^{\leq}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| &> \frac{1}{2}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\| \\
&\geq \frac{1}{2}\|E_{\alpha,j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \\
&\geq \frac{1}{4}\|E_{\alpha,j_0}(\mathbf{v}-\mathbf{v}')\| \\
&\geq \frac{1}{4}\|\pi^{\geq}_{j_1+2}(E_{\alpha,j_0}(\mathbf{v}-\mathbf{v}'))\| \\
&= \frac{\alpha^{j_1-j_0+2}}{4}\|E_{\alpha,j_1+2}(\mathbf{v}-\mathbf{v}')\| \\
&\geq \frac{\alpha^{j_1-j_0+2}}{4}\,\lambda_1(E_{\alpha,j_1+2}(\mathcal{L}))\,,
\end{aligned}$$

where the first inequality follows from (8) and the third inequality uses Lemma 4.9 with lattice $E_{\alpha,j_0}(\mathcal{L})$. On the other hand, we know that $\|\pi^{\leq}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \leq \mu(\mathcal{L}_{j_1})$ according to Lemma 2.2. Then we have

$$\frac{\alpha^{j_1-j_0+2}}{4}\,\lambda_1(E_{\alpha,j_1+2}(\mathcal{L})) < \mu(\mathcal{L}_{j_1}) \leq \frac{q}{\gamma^2}\,\lambda_1(E_{\alpha,j_1+2}(\mathcal{L}))\,, \tag{16}$$

where the last inequality uses Corollary 4.11. Since $\alpha^{j_1-j_0+2} \geq \alpha^3 \geq 1/8$, (16) contradicts the condition $q \leq \gamma^2/32$.

Based on (15), we continue to prove (10) with the following observation:

$$\begin{aligned}
\|E_{\alpha,j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|^2 &= \sum_{i=j_0}^{m} \alpha^{2(i-j_0)}\|\pi_i(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|^2 \\
&\geq \alpha^{2(j_2-j_0)} \sum_{i=j_0}^{j_2} \|\pi_i(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|^2 \\
&= \alpha^{2(j_2-j_0)}(\|\mathbf{x}-\mathbf{y}-\mathbf{v}'\|^2 - \|\pi^{<}_{j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|^2 - \|\pi^{>}_{j_2}(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|^2) \\
&\geq \alpha^{2(j_2-j_0)}(\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2 - \mu^2(\mathcal{L}_{j_0-1}) - \|\pi^{>}_{j_2}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2)\,,
\end{aligned} \tag{17}$$

where the last inequality uses the following three facts: (i) as $\mathbf{v}$ realizes $\mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})$, $\|\mathbf{x}-\mathbf{y}-\mathbf{v}'\| \geq \|\mathbf{x}-\mathbf{y}-\mathbf{v}\|$; (ii) the term $\|\pi^{<}_{j_0}(\mathbf{x}-\mathbf{y}-\mathbf{v}')\|$ is bounded from above by (12); and (iii) $\pi^{>}_{j_2}(\mathbf{x}-\mathbf{y}-\mathbf{v}') = \pi^{>}_{j_2}(\mathbf{x}-\mathbf{y}-\mathbf{v})$ due to (15). Moreover, according to (13) and (9),

$$\begin{aligned}
\mu(\mathcal{L}_{j_0-1}) &\leq \frac{1}{4}\,\lambda_1(E_{\alpha,j_1}(\mathcal{L})) \\
&\leq \frac{1}{2}\|E_{\alpha,j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\| \\
&\leq \frac{1}{2}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|\,,
\end{aligned}$$

and according to (8),

$$\begin{aligned}
\|\pi^{>}_{j_2}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 &\leq \|\pi^{>}_{j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 \\
&< \frac{1}{2}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2\,.
\end{aligned}$$

Hence (17) is further bounded from below by

$$\alpha^{2(j_2-j_0)}\Big(1-\frac{1}{4}-\frac{1}{2}\Big)\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2 \geq \frac{\alpha^4}{4}\|\mathbf{x}-\mathbf{y}-\mathbf{v}\|^2 \geq \frac{\alpha^4}{4}\|\pi_{\overline{\leq} j_1}(\mathbf{x}-\mathbf{y}-\mathbf{v})\|^2 .$$

This completes the proof of (10), and the proof of the lemma. ◀

4.5 Summary of Embedding into Tori

By Lemma 4.3, for any lattice $\mathcal{L}$, there exists an $(n\sqrt{n}, n)$-filtration[2] $\mathcal{F}$ of $\mathcal{L}$. Applying Lemmas 4.8 and 4.12 to the embedding $E_{\mathcal{F},\alpha}$ with $\alpha = 1/2$, we have the following.

▶ **Lemma 4.13.** *For any sufficiently large $n \geq 1$ and lattice $\mathcal{L} \subseteq \mathbb{R}^n$, there exists $m \geq 1$ and embedding $F_{\mathcal{L}} = (F_{\mathcal{L},1}, \ldots, F_{\mathcal{L},m})$ such that each $F_{\mathcal{L},j}$ maps the torus $\mathbb{R}^n/\mathcal{L}$ to some other torus, and $F_{\mathcal{L}}$ satisfies*

$$\sum_{j=1}^{m} \mathrm{dist}_{F_{\mathcal{L},j}(\mathbb{R}^n/\mathcal{L})}(F_{\mathcal{L},j}(\mathbf{x}), F_{\mathcal{L},j}(\mathbf{y}))^2 \leq c_{E,u} \cdot \mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})^2 \text{ , and}$$

$$\sum_{j=1}^{m} \min\big(\mathrm{dist}_{F_{\mathcal{L},j}(\mathbb{R}^n/\mathcal{L})}(F_{\mathcal{L},j}(\mathbf{x}), F_{\mathcal{L},j}(\mathbf{y})), p(n)\,\lambda_1(F_{\mathcal{L},j}(\mathcal{L}))\big)^2 \geq c_{E,l} \cdot \mathrm{dist}_{\mathbb{R}^n/\mathcal{L}}(\mathbf{x},\mathbf{y})^2 ,$$

where $c_{E,u}$ and $c_{E,l}$ are positive absolute constants and $p(n)$ is a fixed polynomial.

5 Putting it All Together

▶ **Theorem 1.1.** *For any lattice $\mathcal{L} \subseteq \mathbb{R}^n$ there exists a metric embedding of $\mathbb{R}^n/\mathcal{L}$ into Hilbert space with distortion $O(\sqrt{n \log n})$.*

Proof. It suffices to show the embedding for sufficiently large n (by, say, using the embedding from [3] for small n). Consider the composition

$$\Big(H^{(k)}_{F_{\mathcal{L},1}(\mathcal{L})} \circ F_{\mathcal{L},1}, \ldots, H^{(k)}_{F_{\mathcal{L},m}(\mathcal{L})} \circ F_{\mathcal{L},m}\Big) ,$$

where $(F_{\mathcal{L},1}, \ldots, F_{\mathcal{L},m})$ is the embedding provided by Lemma 4.13. Let $k = \lceil \log_2 p(n) \rceil + 1$ (where $p(n)$ is the fixed polynomial in Lemma 4.13). By Lemma 3.6 and Lemma 4.13, noting that the modified contraction properties in both match, it follows immediately that the composed embedding has distortion at most

$$\sqrt{\frac{\pi k n \cdot c_{E,u}}{c_H \cdot c_{E,l}}} ,$$

where c_H, $c_{E,u}$ and $c_{E,l}$ are all absolute constants. Note that $k = \Theta(\log n)$. Hence the distortion of the composed embedding is $O(\sqrt{n \log n})$. ◀

[2] This choice of filtration actually only shows the Lemma 4.13 for sufficiently large n. Choosing a $(32n\sqrt{n}, 32n)$-filtration gives us the lemma for all $n \geq 1$.

References

1 Ishay Haviv and Oded Regev. The Euclidean distortion of flat tori. *J. Topol. Anal.*, 5(2):205–223, 2013. Preliminary version in APPROX 2010. doi:10.1142/S1793525313500064.

2 Piotr Indyk. Algorithmic applications of low-distortion geometric embeddings. In *42nd IEEE Symposium on Foundations of Computer Science (Las Vegas, NV, 2001)*, pages 10–33. IEEE Computer Soc., Los Alamitos, CA, 2001.

3 Subhash Khot and Assaf Naor. Nonembeddability theorems via Fourier analysis. *Math. Ann.*, 334(4):821–852, 2006. Preliminary version in FOCS 2005. doi:10.1007/s00208-005-0745-0.

4 John Milnor and Dale Husemoller. *Symmetric bilinear forms*. Springer-Verlag, New York-Heidelberg, 1973. Ergebnisse der Mathematik und ihrer Grenzgebiete, Band 73.

A Tight $(3/2 + \varepsilon)$ Approximation for Skewed Strip Packing

Waldo Gálvez
Technical University of Munich, Germany
galvez@in.tum.de

Fabrizio Grandoni
IDSIA, USI-SUPSI, Manno, Switzerland
fabrizio@idsia.ch

Afrouz Jabal Ameli
IDSIA, USI-SUPSI, Manno, Switzerland
afrouz@idsia.ch

Klaus Jansen
University of Kiel, Germany
kj@informatik.uni-kiel.de

Arindam Khan
Indian Institute of Science, Bangalore, India
arindamkhan@iisc.ac.in

Malin Rau
Univ. Grenoble Alpes, CNRS, Inria, Grenoble INP*, LIG, Grenoble, France
malin.rau@inria.fr

Abstract

In the *Strip Packing* problem, we are given a vertical half-strip $[0, W] \times [0, +\infty)$ and a collection of open rectangles of width at most W. Our goal is to find an axis-aligned (non-overlapping) packing of such rectangles into the strip such that the maximum height OPT spanned by the packing is as small as possible. Strip Packing generalizes classical well-studied problems such as *Makespan Minimization* on identical machines (when rectangle widths are identical) and *Bin Packing* (when rectangle heights are identical). It has applications in manufacturing, scheduling and energy consumption in smart grids among others. It is NP-hard to approximate this problem within a factor $(3/2 - \varepsilon)$ for any constant $\varepsilon > 0$ by a simple reduction from the *Partition* problem. The current best approximation factor for Strip Packing is $(5/3+\varepsilon)$ by Harren et al. [Computational Geometry '14], and it is achieved with a fairly complex algorithm and analysis.

It seems plausible that Strip Packing admits a $(3/2 + \varepsilon)$-approximation. We make progress in that direction by achieving such tight approximation guarantees for a special family of instances, which we call *skewed instances*. As standard in the area, for a given constant parameter $\delta > 0$, we call *large* the rectangles with width at least δW and height at least δOPT, and *skewed* the remaining rectangles. If all the rectangles in the input are large, then one can easily compute the optimal packing in polynomial time (since the input can contain only a constant number of rectangles). We consider the complementary case where all the rectangles are skewed. This second case retains a large part of the complexity of the original problem; in particular, it is NP-hard to approximate within a factor $(3/2 - \varepsilon)$ and we provide an (almost) tight $(3/2 + \varepsilon)$-approximation algorithm.

2012 ACM Subject Classification Theory of computation → Packing and covering problems

Keywords and phrases strip packing, approximation algorithm

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.44

Category APPROX



© Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau;
licensed under Creative Commons License CC-BY

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 44; pp. 44:1–44:18



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

Funding *Fabrizio Grandoni*: Partially supported by the SNSF Excellence Grant 200020B_182865/1.
Afrouz Jabal Ameli: Partially supported by the SNSF Excellence Grant 200020B_182865/1.
Malin Rau: Partially supported by the French research program ENERGUMEN ANR-18-CE25-0008.

1 Introduction

In this paper, we consider the *Strip Packing* problem, a well-studied classical rectangle packing problem (see Section 2 for a formal definition). Here, we are given a vertical half-strip of (integral) width W, plus a collection $\mathcal{R}$ of rectangles of width at most W. Our objective is to find an axis-aligned packing of $\mathcal{R}$ (where rectangles do not overlap) such that the maximum height spanned by the packing is as small as possible.

Strip Packing generalizes famous problems. For example, it generalizes Makespan Minimization (on identical machines) [11] when all the rectangle widths are 1 (here W would be the number of processors), and also generalizes Bin Packing [12] when all the rectangle heights are 1 (here the height OPT of the optimal solution would be the optimal number of bins). Strip Packing has several natural applications. For example, there are many manufacturing settings where rectangular pieces have to be cut out of some roll of raw material while using a rectangular piece of that roll of minimum length. Another application is the minimization of the peak energy consumption in smart-grids [28, 37]: here heights and widths model the energy consumption and duration, respectively, of a given set of jobs. For analogous reasons, it captures scenarios where a given set of jobs needs to be allocated a *consecutive* amount of a given resource (memory locations, frequencies, etc.) for a given amount of time.

Strip Packing is strongly NP-hard [17], and hence it is reasonable to consider approximation algorithms for it. A simple reduction from the *Partition* problem shows that it is not possible to obtain a $(\frac{3}{2} - \varepsilon)$-approximation algorithm (with polynomial running time) for any $\varepsilon > 0$ unless P=NP (more details on this reduction are given later). The first non-trivial approximation algorithm for Strip Packing, with approximation ratio 3, was given by Baker, Coffman, and Rivest [4]. The First-Fit-Decreasing-Height algorithm (FFDH) by Coffman et al. [27] gives a 2.7-approximation. Sleator [35] gave an algorithm that generates a packing of height $2OPT + \frac{h_{max}}{2}$, where h_{max} is the maximum height of a rectangle in the instance, hence achieving a 2.5-approximation. Afterwards, Steinberg [36] and Schiermeyer [34] independently improved the approximation ratio to 2. Harren and van Stee [19] first broke the barrier of 2 with their 1.9396-approximation. The present best $(\frac{5}{3} + \varepsilon)$-approximation is due to Harren et al. [18].

The Strip Packing problem has also been studied in the pseudopolynomial setting, i.e., when $N = n^{O(1)}$. After a series of recent improvements [33, 1, 16, 20, 23], Jansen and Rau [22] have given a pseudopolynomial time algorithm with an almost tight $(\frac{5}{4} + \varepsilon)$-approximation ratio.

In terms of asymptotic approximations, the barrier of $\frac{3}{2}$ can also be beaten. The best results in these terms are an AFPTAS presented by Kenyon and Rémila [29] which produces a solution of height $(1 + \varepsilon)OPT + O\left(\frac{h_{max}}{\varepsilon^2}\right)$, and an APTAS which generates a solution of height $(1 + \varepsilon)OPT + h_{max}$ by Jansen and Solis-Oba [24]. For the variant of Strip Packing *with Rotations*, where the rectangles are allowed to be rotated by 90 degrees, Jansen and van Stee [25] gave an APTAS (see also [13, 32] for related results).

1.1 Related Work

Strip Packing has rich connections with many other important geometric packing problems such as Two-dimensional Bin Packing (2BP) and Two-dimensional Geometric Knapsack (2GK). In 2BP, we are given a set of rectangles and unit square bins, and the goal is to pack

all the rectangles into minimum number of bins. The problem is known to be APX-hard [6] and the present best approximation ratio is 1.405 [7]. In 2GK, we are given a set of rectangles (with associated profit) and unit square knapsack, and the goal is to pack a subset of rectangles into the knapsack maximizing the total profit. This problem is strongly NP-hard even when all items are squares with unit profit [31]. The present-best approximation ratio is 1.89 due to Gálvez et al. [14] (see also [3, 26]).

Strip Packing has also been well studied for higher dimensions. The present best asymptotic approximation for 3-D Strip Packing is due to Jansen and Prädel [21] who presented a 1.5-approximation extending techniques from 2-D Bin Packing.

Another related problem is the *Independent Set of Rectangles* problem: here we are given a collection of axis-parallel rectangles embedded in the plane, and we need to find a maximum cardinality/weight subset of non-overlapping rectangles [2, 8, 9].

We refer the readers to [10, 30] for surveys on geometric packing problems.

1.2 Our Contribution

In this paper, we study a special case of Strip Packing, where all rectangles are *skewed*. In more detail, we say that a rectangle R is δ-*large* if, for some fixed constant $\delta > 0$, its width is at least a δ fraction of the width W of the strip and its height is at least a δ fraction of the height OPT of the optimal packing; otherwise, the rectangle is δ-*skewed*. We just say that a rectangle is large or skewed when δ is clear from the context. An instance of Strip Packing is δ-skewed if all the rectangles in the input are such.

This special case is non-trivial: in particular, the mentioned $3/2 - \varepsilon$ hardness of approximation holds also for this special case with minor adaptations (see Section 5). We also believe that this special case is practically relevant: e.g., it captures scenarios where no job can consume a significant amount of the global resource (energy, memory space, etc.) for a significant amount of time. Our main result is as follows (see Sections 3-4).

▶ **Theorem 1.** *For a given constant $\varepsilon' > 0$ and a small enough positive constant $\delta \leq (\varepsilon')^{(1/\varepsilon')^{O(1)}}$, there exists a polynomial-time $\left(\frac{3}{2} + \varepsilon'\right)$-approximation algorithm for δ-skewed Strip Packing.*

We remark that our algorithm does not need to recognize first if the instance is δ-skewed: It always returns a feasible solution, but only if the instance satisfies the requirements, its approximation ratio is guaranteed.

Our result suggests that, in order to obtain a better (possibly $3/2 + \varepsilon$) approximation for the general case of Strip Packing, one of the main obstacles is the interaction between large and skewed rectangles.

1.2.1 Organization

In Section 2, we introduce some useful notation and preliminary results. In Section 3, we prove the existence of a good enough packing with certain structural properties. The mentioned structure is exploited to derive an algorithm with the claimed approximation guarantee in Section 4. Section 5 contains our hardness of approximation result.

2 Preliminaries

A Strip Packing instance consists of a vertical strip of integral width W in the two-dimensional plane, i.e. $[0, W] \times \mathbb{R}_{\geq 0}$, and a set of open rectangles $\mathcal{R}$, where each rectangle $R \in \mathcal{R}$ is characterized by its integral height $h(R)$ and integral width $w(R)$. An embedding of $\mathcal{R}$ is

given by specifying a bottom-left position $(x(R), y(R))$ for each $R \in \mathcal{R}$. The interpretation is that R is embedded in the plane in the region $\big(x(R), x(R)+w(R)\big) \times \big(y(R), y(R)+h(R)\big)$. An embedding is a feasible packing into the strip if the following two conditions hold: (1) each $R \in \mathcal{R}$ is embedded inside the strip, namely $0 \leq x(R) \leq W - w(R)$ and $y(R) \geq 0$ and (2) rectangles do not overlap, namely, for any two $R, S \in \mathcal{R}$, $\Big(\big(x(R), x(R)+w(R)\big) \times \big(y(R), y(R)+h(R)\big)\Big) \cap \Big(\big(x(S), x(S)+w(S)\big) \times \big(y(S), y(S)+h(S)\big)\Big) = \emptyset$. The height of a feasible packing is the maximum height spanned by any embedded rectangle, namely the maximum value of $y(R)+h(R)$. The goal of Strip Packing is to compute a feasible packing of minimum height OPT. W.l.o.g. we can restrict our attention to packings where the coordinates $(x(R), y(R))$ are integral as any feasible packing can be transformed into a feasible packing satisfying this property (intuitively, by pushing rectangles to the bottom-left as much as possible while keeping feasibility).

Given a subset of rectangles $\mathcal{S} \subseteq \mathcal{R}$, we denote by $w(\mathcal{S}) := \sum_{R \in \mathcal{S}} w(R)$, $h(\mathcal{S}) := \sum_{R \in \mathcal{S}} h(R)$, and $a(\mathcal{S}) := \sum_{R \in \mathcal{S}} h(R)w(R)$ the total width, height, and area of $\mathcal{S}$, respectively. The operation of changing the bottom-left corner of a rectangle R in a given packing from $(x(R), y(R))$ to $(x(R), y(R)+a)$ will be denoted by *shifting R vertically by a*. Analogously, changing the bottom-left coordinate from $(x(R), y(R))$ to $(x(R)+a, y(R))$ will be denoted by *shifting R horizontally by a*. These operations are only allowed if the resulting packing is feasible.

A *box* of size $w \times h$ denotes a rectangular region of width w and height h. We sometimes embed boxes into the strip similarly to rectangles. A monotone polygonal chain is a curve specified by a sequence of points $(A_1, A_2, \ldots, A_n)$ called its vertices. The curve itself consists of the line segments connecting the consecutive vertices, and we require that the x-coordinates of points A_i are non-decreasing and the segments are horizontal or vertical. We say that a rectangle R in the packing *lies above (resp. below)* one such P if for any $x_1 \in (x(R), x(R)+w(R))$ we have that $y(R)$ (resp. $y(R)+h(R)$) is not smaller (resp. not larger) than the largest (resp., smallest) y-coordinate of P at x-coordinate x_1.

We can assume w.l.o.g. that W is lower bounded by a sufficiently large constant, in particular $W \geq 1/\varepsilon$. Otherwise one easily obtains a PTAS for δ-skewed instances[1].

2.1 Next Fit Decreasing Height

One of the most recurring tools used as a subroutine in countless results on geometric packing problems is the Next Fit Decreasing Height (NFDH) algorithm, which was originally analyzed in [27]. We will use a variant of this algorithm to pack rectangles inside a rectangular box and analyze its properties. We provide a full proof for the sake of completeness.

Suppose we are given a box C of size $w \times h$, and a set of rectangles $\mathcal{R}'$ each one fitting in the box. NFDH computes in polynomial time a packing of $\mathcal{R}'' \subseteq \mathcal{R}'$ as follows. It sorts the rectangles $R \in \mathcal{R}'$ in non-increasing order of height $h(R)$, and considers rectangles in that order $R_1, \ldots, R_n$. Then the algorithm works in rounds $j \geq 1$. At the beginning of round j it is given an index $n(j)$ and a horizontal segment $L(j)$ going from the left to the right side of C. Initially $n(1) = 1$ and $L(1)$ is the bottom side of C. In round j the algorithm packs a maximal set of rectangles $R_{n(j)}, \ldots, R_{n(j+1)-1}$, with bottom side touching $L(j)$ one next to the other from left to right (a *shelf*). The segment $L(j+1)$ is the horizontal segment

[1] Choosing δ such that $\delta W < 1$ enforces each rectangle to have height at most δOPT (otherwise it would be large). A PTAS follows, e.g., from [24].

containing the top side of $R_{n(j)}$ and ranging from the left to the right side of C. The process halts at round r when either all rectangles have being packed or $R_{n(r+1)}$ does not fit above $R_{n(r)}$.

We prove the following standard result.

▶ **Lemma 2** ([27]). *Let C be a given box of size $w \times h$ and $\mathcal{R}' \subseteq \mathcal{R}$. Assume that, for some given parameter $\varepsilon' \in (0,1)$, for each $R \in \mathcal{R}'$ one has $w(R) \leq \varepsilon' w$ and $h(R) \leq \varepsilon' h$. Then NFDH is able to pack in C a subset $\mathcal{R}'' \subseteq \mathcal{R}'$ of area at least $a(\mathcal{R}'') \geq \min\{a(\mathcal{R}'), (1-2\varepsilon')w \cdot h\}$. In particular, if $a(\mathcal{R}') \leq (1-2\varepsilon')w \cdot h$, all rectangles in $\mathcal{R}'$ are packed.*

Proof. The claim trivially holds if all rectangles are packed. Thus suppose that this is not the case. Observe that $\sum_{j=1}^{r+1} h(R_{n(j)}) > h$, otherwise rectangle $R_{n(r+1)}$ would fit in the next shelf above $R_{n(r)}$; hence $\sum_{j=2}^{r+1} h(R_{n(j)}) > h - h(R_{n(1)}) \geq (1-\varepsilon')h$. Observe also that the total width of rectangles packed in each round j is at least $w - \varepsilon' w = (1-\varepsilon')w$, since $R_{n(j+1)}$, of width at most $\varepsilon' w$, does not fit to the right of $R_{n(j+1)-1}$. It follows that the total area of the rectangles packed in round j is at least $(w-\varepsilon' w)h(R_{n(j+1)-1})$, and thus

$$a(\mathcal{R}'') \geq \sum_{j=1}^{r} (1-\varepsilon')w \cdot h(R_{n(j+1)-1}) \geq (1-\varepsilon')w \sum_{j=2}^{r+1} h(R_{n(j)}) \geq (1-\varepsilon')^2 w \cdot h \geq (1-2\varepsilon')w \cdot h. \quad \blacktriangleleft$$

2.2 Container Packings

Similar to recent work on related problems (e.g., [14, 5]), we will exploit a *container-based* packing approach. The idea is to partition the solution into a constant number of axis-aligned rectangular regions (*containers*). The sizes (and therefore positions) of these containers can be *guessed* in polynomial time, and subsequently, rectangles are packed inside the containers in a simple way: either one next to the other from left to right (*vertical container*), or one on top of the other from bottom to top (*horizontal container*), or by means of NFDH (*area container*). We further require that the rectangles R packed into an area container of size $w \times h$ satisfy $w(R) \leq \varepsilon' w$ and $h(R) \leq \varepsilon' h$ for a constant $\varepsilon' > 0$ to be fixed later. We call this an ε'-area container.

We will make use of the following standard PTAS to pack rectangles into a constant number of containers. The basic idea is to reduce the problem to an instance of the Maximum Generalized Assignment Problem (GAP) with one bin per container, and then use a PTAS for the latter problem plus NFDH to repack rectangles in area containers. We recall that in GAP, we are given a collection of n items and a set of k (one-dimensional) bins, each one characterized by a positive size. Each item has a profit[2] and a positive size per bin (possibly different for different bins). Our goal is to compute a maximum profit subset of items and an assignment of them into the bins so that the total size of items packed in each bin is at most the size of the bin. GAP admits a PTAS for constant k (see e.g. Section E.2 in [15]) and the following lemma shows how to use it to pack the rectangles into a given set of containers.

▶ **Lemma 3.** *For any constant $\varepsilon' > 0$, given a set of rectangles $\mathcal{R}$ that can be packed into a given set of k containers (each container being either vertical, horizontal or ε'-area), k constant, there is an algorithm to pack $\mathcal{R}' \subseteq \mathcal{R}$ with $a(\mathcal{R}') \geq (1-3\varepsilon')a(\mathcal{R})$ into the mentioned containers.*

[2] The same item might have different profit on different knapsacks; however, we do not need this extension here.

Proof. We let $w(C_j) \times h(C_j)$ be the size of the j-th container C_j. We build an instance of GAP as follows. There is one item R per rectangle $R \in \mathcal{R}$, with profit $a(R)$. For each horizontal container C_j, we create a knapsack j of size $S_j := h(C_j)$. Furthermore, we define the size $s(R,j)$ of rectangle R w.r.t. knapsack j as $h(R)$ if $h(R) \leq h(C_j)$ and $w(R) \leq w(C_j)$. Otherwise $s(R,j) = +\infty$ (meaning that R does not fit in C_j). The construction for vertical containers is symmetric. For each area container C_j we create a knapsack j of size $S_j = a(C_j)$ and define the size $s(R,j)$ of rectangle R w.r.t. knapsack j as $a(R)$ if $h(R) \leq \varepsilon' h(C_j)$ and $w(R) \leq \varepsilon' w(C_j)$, setting $b(R,j) = +\infty$ otherwise (meaning that the rectangle is not small enough with respect to the dimensions of the container).

We next apply the mentioned PTAS for GAP to this instance, so as to obtain a solution $\mathcal{R}''$ to GAP of profit at least $a(\mathcal{R}'') \geq (1-\varepsilon')a(\mathcal{R})$. We build a feasible packing of $\mathcal{R}' \subseteq \mathcal{R}''$ into the containers as follows. Let $\mathcal{R}_j$ be the items packed into knapsack j. If C_j is vertical, we pack rectangles $\mathcal{R}_j$ into this container bottom-most and from left to right one next to the other in any order. By definition all rectangles $\mathcal{R}_j$ will fit. A symmetric construction works if C_j is horizontal. If C_j is area, we pack a subset $\mathcal{R}'_j$ of $\mathcal{R}_j$ into it using NFDH. By Lemma 2, we either have $\mathcal{R}'_j = \mathcal{R}_j$, or it must be the case that $a(\mathcal{R}'_j) \geq (1-2\varepsilon')w(C_j)h(C_j) = (1-2\varepsilon')a(C_j)$. Consider the second case. Let $\mathcal{R}''_j = \mathcal{R}_j \setminus \mathcal{R}'_j$ be the rectangles which are not packed. Observe that $a(\mathcal{R}_j) \leq a(C_j)$ by the feasibility of the GAP solution, hence

$$\frac{a(\mathcal{R}''_j)}{a(\mathcal{R}_j)} = 1 - \frac{a(\mathcal{R}'_j)}{a(\mathcal{R}_j)} \leq 1 - \frac{(1-2\varepsilon')a(C_j)}{a(\mathcal{R}_j)} \leq 2\varepsilon'.$$

Thus altogether $a(\mathcal{R}') \geq a(\mathcal{R}'')(1-2\varepsilon') \geq a(\mathcal{R})(1-2\varepsilon')(1-\varepsilon') \geq a(\mathcal{R})(1-3\varepsilon')$. ◀

Notice that the containers may have considerable free space inside, but the lemma just claims that the total area of the rectangles that the algorithm is not packing is negligible. Whenever this lemma is applied, we will pack the remaining rectangles into an extra rectangular box of small area and carefully argue where to place it.

2.3 Classification of Rectangles

From now on, we will assume that instance $(\mathcal{R}, W)$ is δ-skewed for some $\delta > 0$ to be fixed later. By OPT, we denote both the optimal height and an optimal packing: the meaning will be clear from the context. We can assume that OPT is even (otherwise, multiply heights by a factor 2).

We will assume that our algorithm is given in the input a value OPT' such that $OPT \leq OPT' \leq (1+\varepsilon)OPT$. This assumption can be removed as follows. We compute, say, a 2-approximation APX by Steinberg's algorithm for Strip Packing and then run our algorithm for all the (constantly many) values $OPT' = (1+\varepsilon)^j \frac{APX}{2(1+\varepsilon)}$ which fit in the range $[\frac{APX}{2(1+\varepsilon)}, APX(1+\varepsilon)]$. One of these values will satisfy the claim. In order to keep the notation light, we simply use OPT to denote this value OPT'. Therefore, all the approximation factors should be scaled by a factor $(1+\varepsilon)$ in order to consider the true value of OPT.

Let $\varepsilon > 0$ and assume for simplicity that $\frac{1}{\varepsilon} \in \mathbb{N}$. We will classify the rectangles according to their heights as follows:

- The set of *tall* rectangles $\mathcal{T} = \{R \in \mathcal{R} : h(R) > \frac{1}{2}OPT\}$;
- The set of *vertical* rectangles $\mathcal{V} = \{R \in \mathcal{R} : h(R) \in (\delta OPT, \frac{1}{2}OPT]\}$;
- The set of *short* rectangles $\mathcal{S} = \{R \in \mathcal{R} : h(R) \leq \delta OPT\}$;

2.4 Linear Grouping

We need the following lemma whose proof (given in the appendix) is based on *linear grouping*, a standard technique in the area of rectangle packing. Given a subset $\mathcal{S}$ of rectangles, let $\mathcal{S}_{hslice}$ be the set of rectangles obtained by taking each $R \in \mathcal{S}$ and replacing it with $h(R)$ rectangles of height 1 and width $w(R)$ (*horizontal slices*). We define symmetrically the set $\mathcal{S}_{vslice}$ of *vertical slices*. Notice that any embedding of $\mathcal{S}$ naturally induces an embedding of $\mathcal{S}_{hslice}$ and $\mathcal{S}_{vslice}$.

▶ **Lemma 4.** *Let $\varepsilon' > 0$ be a given constant, $\mathcal{P}$ be a rectangular region of size $W \times H$ and $\mathcal{H}$ be a subset or rectangles of height at most $\delta \cdot H$ each for some constant $\delta \in [1, 0)$. Suppose that $\mathcal{H}_{hslice}$ can be packed into a set $\mathcal{B}$ of $K = O_{\varepsilon'}(1)$ boxes contained in $\mathcal{P}$. Then, for $\delta \leq (\varepsilon'/K)^{(K/\varepsilon')^{O(1)}}$, there exists a partition of $\mathcal{H}$ into two sets $\mathcal{H}^{cont}$ and $\mathcal{H}^{disc}$ such that:*

1. *$\mathcal{H}^{cont}$ can be packed into a set of at most $K' = O_{\varepsilon'}(1)$ horizontal and ε'-area containers, where each container is fully contained in some box in $\mathcal{B}$.*
2. *$\mathcal{H}^{disc}$ can be packed in one horizontal container of size $\max_{R \in \mathcal{H}}\{w(R)\} \times (\varepsilon')^2 H$ and one ε'-area container of size $\varepsilon' W \times \varepsilon' H$.*
3. *The sizes of the above containers belongs to a set that can be computed in polynomial time.*

A symmetric claim holds for a subset of rectangles $\mathcal{V}'$ of width at most $\delta \cdot W$ such that $\mathcal{V}'_{vslice}$ can be packed into the corresponding boxes.

3 Existence of a Structured Solution

In this section, we will prove our main structural result.

▶ **Theorem 5.** *For any given constant $\varepsilon > 0$ and any given instance of δ-skewed Strip Packing $(\mathcal{R}, W)$ with $\delta = \Omega_\varepsilon(1)$ small enough, there exists a feasible container packing such that the following holds:*

1. *The total height of the packing is $\left(\frac{3}{2} + O(\varepsilon)\right) OPT$.*
2. *The number of containers is $O_\varepsilon(1)$ and their possible sizes belong to a set that can be computed in polynomial time.*
3. *Given any fixed ordering of $\mathcal{T}$ in non-increasing order of height, $\mathcal{T}$ can be partitioned into subsequences each one fitting in precisely one vertical container.*
4. *The packing leaves a free rectangular region (**free box**) of size $\varepsilon^2 W \times \frac{1}{2} OPT$.*

To achieve the above result, we proceed in three steps:

1. We describe a packing of $\mathcal{T} \cup \mathcal{S}_{hslice}$ with height at most $(3/2 + O(\varepsilon))OPT$ (see Section 3.1) into $O_\varepsilon(1)$ boxes. This packing leaves a free space of at least $(1/2 + \Omega(\varepsilon))OPT + a(\mathcal{V})$.
2. We describe (see Section 3.2) how to pack $\mathcal{V}_{vslice}$ within the free space of the previous packing using $O_\varepsilon(1)$ extra boxes. Furthermore, we guarantee that there is a free box (not containing any rectangle) of size at least $\Omega(\varepsilon)W \times \frac{1}{2}OPT$. Guaranteeing the latter property is critical, and it is the main technical novelty in our approach.
3. Finally (see Section 3.3), we convert the above packing into a feasible container packing (via Lemma 4) inside the above boxes. The residual containers that do not fit into the boxes can be placed inside the free box (still leaving enough space) plus a new box of size $W \times O(\varepsilon^2)OPT$ that can be placed on top of the previous packing.

The reason for leaving a free box will be clearer in Section 4, where we will describe our final algorithm.

3.1 Packing of $\mathcal{T} \cup \mathcal{S}_{hslice}$

In this section, we describe a packing of $\mathcal{T} \cup \mathcal{S}_{hslice}$. The proof of the following Lemma is illustrated in Figure 1.

▶ **Lemma 6.** *For any given constant $\varepsilon \in (0, 1/4]$ with $1/\varepsilon$ integral and $\delta = \Omega_\varepsilon(1)$ sufficiently small, it is possible to pack $\mathcal{T} \cup \mathcal{S}_{hslice}$ into the region $\mathcal{P} = [0, W] \times [0, (\frac{3}{2} + 15\varepsilon)OPT]$ in such a way that:*

1. *Rectangles in $\mathcal{T}$ are packed into at most $1/\varepsilon$* vertical boxes*, slices in $\mathcal{S}_{hslice}$ are packed into at most $1/\varepsilon + 1$* horizontal boxes*, and the remaining area is partitioned into at most $2/\varepsilon$* free boxes*. Furthermore, given any fixed ordering of $\mathcal{T}$ in non-increasing order of height, it is possible to partition $\mathcal{T}$ into subsequences such that each subsequence fits into precisely one vertical box.*
2. *The sizes of the boxes belong to a set that can be computed in polynomial time.*
3. *The total area of the free boxes is at least $(\frac{1}{2} + 9\varepsilon)OPT \cdot W + a(\mathcal{V})$.*

Proof. Consider the embedding of $\mathcal{T} \cup \mathcal{S}_{hslice}$ induced by the optimum solution. Let us draw the horizontal line $y = \frac{1}{2}OPT$ and partition $\mathcal{S}_{hslice}$ into two sets $\mathcal{S}^{top}_{hslice}$ and $\mathcal{S}^{bottom}_{hslice}$ corresponding to the rectangles in $\mathcal{S}_{hslice}$ which are packed above and below the line $y = \frac{1}{2}OPT$ respectively (notice that this line does not intersect any rectangle in S_{hslice} as OPT is even by assumption). If we shift up rectangles in $\mathcal{S}^{bottom}_{hslice}$ by OPT, we obtain a feasible packing (since the region $[0, W] \times [OPT, \frac{3}{2}OPT]$ was empty) with final height at most $\frac{3}{2}OPT$. Notice that every rectangle in $\mathcal{T}$ intersects the horizontal segment $[0, W] \times \left\{\frac{1}{2}\right\}$.

Now, let us shift down each rectangle R in $\mathcal{T}$ so that its bottom coordinate becomes zero (again, the packing remains feasible). Next, we shift rectangles so that the ones in $\mathcal{T}$ appear one next to the other in the bottom left part of the packing, in non-increasing order of height. To this aim, we proceed recursively as follows. Let $T_1, \ldots, T_q$ be the considered ordering of $\mathcal{T}$ in non-increasing order of height. At the beginning of iteration $i \geq 1$, we are given a feasible packing where $T_1, \ldots, T_{i-1}$ are packed from left to right one next to the other as left as possible (and with bottom coordinate 0). We consider the region $A_i := [w_{i-1}, W] \times [0, h(T_i)]$, where $w_{i-1} = \sum_{j \leq i-1} w(T_j)$. Let L_i be the portion of A_i to the left of (the current embedding of) T_i. Note that every rectangle is either completely contained in or disjoint from L_i since T_i is the tallest rectangle contained in A_i and T_{i-1} is taller than T_i. We move T_i so that its left coordinate is w_{i-1}, and shift L_i to the right by $w(T_i)$, moving consistently all rectangles in L_i. Obviously the new packing satisfies the invariant for the next iteration. At the end of iteration q the packing satisfies the claim.

In the next step, we partition the area not occupied by $\mathcal{T}$ into unit-height stripes. Notice that each rectangle in $\mathcal{S}_{hslice}$ is fully contained in some stripe. We need (for a reason that will be clearer later) to temporarily discard, meaning that we remove them from the packing, some slices as follows. Let us say that a slice is *wide* if its width is at least εW, and *narrow* otherwise. Consider the slices $\mathcal{S}'_{hslice}$ in a given horizontal stripe. This set contains at most $1/\varepsilon$ wide slices. Let w' be the total width of the remaining narrow slices and let $w'' \leq w'$ be the largest multiple of εW. We discard a minimal subset of narrow slices so that the remaining ones have width at most w''. We let $\mathcal{S}^{disc}_{hslice}$ be the set of discarded slices, and $\mathcal{S}^{sel}_{hslice}$ be the remaining (selected) slices.

Next, we push slices $\mathcal{S}^{sel}_{hslice}$ as right as possible. Afterward, we permute the y-coordinates of slices in pairs of stripes so that stripes are sorted from top to bottom in non-increasing order of the total width of the slices contained in them. Observe that this cannot create any conflict with $\mathcal{T}$ (hence the packing remains feasible).

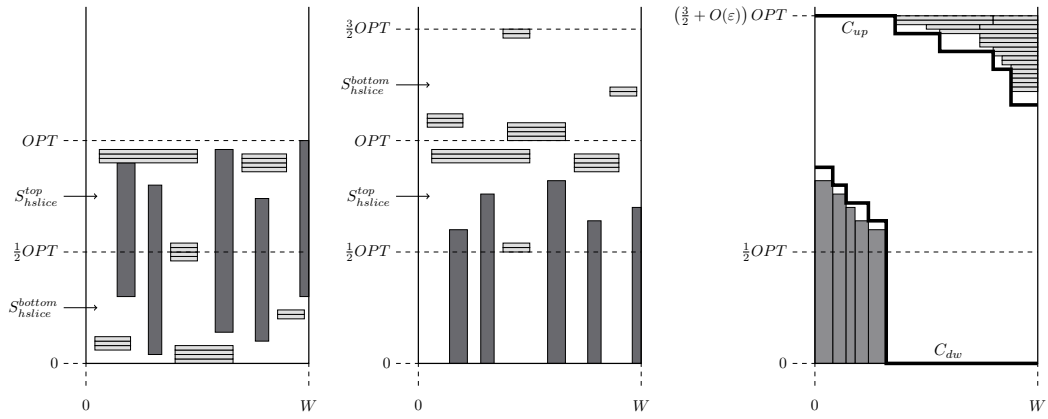


Figure 1 Depiction of the proof of Lemma 6. **Left:** Packing of $\mathcal{T} \cup \mathcal{S}_{hslice}$ in the optimal solution. Light gray rectangles correspond to $\mathcal{S}_{hslice}$, dark gray rectangles correspond to $\mathcal{T}$. **Center:** By shifting $\mathcal{S}^{bottom}_{hslice}$ to the top we can shift down the rectangles in $\mathcal{T}$. **Right:** We can shift now horizontally rectangles in $\mathcal{T}$ and sort stripes. Finally one obtains C_{up} and C_{dw}.

We proceed as follows. We shift up $\mathcal{S}^{sel}_{hslice}$ by $11\varepsilon \cdot OPT$ and construct a polygonal chain C_{up} with the following procedure. The chain starts at coordinate $p^{up}_0 = (0, (\frac{3}{2} + 11\varepsilon)OPT)$. We extend the chain to the right (possibly by a zero amount) until the chain hits some rectangle in $\mathcal{S}^{sel}_{hslice}$. We denote this point by $(x_1, (\frac{3}{2} + 11\varepsilon)OPT)$ and extend the chain down by εOPT, hence reaching some point $p^{up}_1 = (x_1, (\frac{3}{2} + 10\varepsilon)OPT)$. We continue from p^{up}_1 in the same fashion. The procedure ends when the chain reaches the x-coordinate W. Observe that, by construction, every rectangle in $\mathcal{S}^{sel}_{hslice}$ lies above C_{up}. Furthermore, C_{up} is defined by axis parallel segments only and at most $1/\varepsilon$ of them.

Afterward, we build symmetrically a polygonal chain C_{dw} as follows. We start from $p^{dw}_0 = (W, 0)$ and extend the chain to the left (possibly by a zero amount) until we hit some rectangle R in $\mathcal{T}$. Then, we extend the chain up by εOPT, hence reaching a point $p^{dw}_1 = (x_1, \varepsilon OPT)$. We continue from p^{dw}_1 in the same fashion until we reach the x-coordinate 0. Notice that each rectangle from $\mathcal{T}$ lies below C_{dw}. Notice also that C_{dw} is defined by axis parallel segments only and at most $1/\varepsilon$ of them. Furthermore, it is fully below C_{up}. To see the latter, take any coordinate $x \in (0, W)$ (which, for simplicity, is not the position of a vertex of any one of the two chains). Let y_{up} and y_{dw} be the corresponding y-coordinates in C_{up} and C_{dw}, resp. Suppose by contradiction that $y_{dw} > y_{up}$. Observe that by construction the segment $x \times (y_{up}, y_{up} + \varepsilon OPT)$ must hit some rectangle R_{up} in $\mathcal{S}^{sel}_{hslice}$. Symmetrically, the segment $x \times (y_{dw} - \varepsilon OPT, y_{dw})$ must hit some rectangle R_{dw} in $\mathcal{T}$. This however implies that R_{up} and R_{dw} overlapped before the shifting up of $\mathcal{S}^{sel}_{hslice}$ by $11\varepsilon OPT$, a contradiction[3].

We claim that points p^{up}_i and p^{dw}_i have coordinates that belong to a set that can be computed in polynomial time. Recall that OPT is known to the algorithm. Notice that the y coordinates of the points p^{up}_i and p^{dw}_i are multiples of εOPT, hence they satisfy the requirement. Since we know the precise packing of $\mathcal{T}$, we can compute the x-coordinates of points p^{dw}_i explicitly. The x-coordinates of points p^{up}_i have value W minus the sum w'' of the widths of the slices in a given strip. By the previous discarding procedure, w'' is the sum of up to $1/\varepsilon$ widths of input rectangles, plus a multiple of εW. Hence we can compute the set of the possible coordinates in polynomial time.

[3] A shifting up by $2\varepsilon OPT$ would be sufficient to achieve a contradiction here. The extra shift by $9\varepsilon OPT$ is used to create some more free space that is needed in the following arguments.

Let us subdivide the area in the strip between C_{up} and $[0,W]\times(\frac{3}{2}+11\varepsilon)OPT$ by extending to the right the horizontal segments in C_{up}. This gives up to $1/\varepsilon$ boxes $\mathcal{B}_{up}$ that fully contain $\mathcal{S}^{sel}_{hslice}$. Symmetrically, we can subdivide the area in the strip between C_{dw} and $[0,W]$ by extending down the vertical segments in C_{dw}. This provides up to $1/\varepsilon$ boxes $\mathcal{B}_{dw}$ that fully contain $\mathcal{T}$. Next, consider the free area between $\mathcal{B}_{up}$ and $\mathcal{B}_{dw}$. By extending down the vertical sides of the boxes in $\mathcal{B}_{up}$ until reaching $\mathcal{B}_{dw}$ and symmetrically extending up the vertical sides of the boxes in $\mathcal{B}_{dw}$ until reaching $\mathcal{B}_{up}$, we obtain a partition of the free area into up to $2/\varepsilon$ free boxes $\mathcal{B}_{free}$. By the previous discussion, the possible sizes of all the mentioned boxes can be computed in polynomial time.

It remains to pack $\mathcal{S}^{disc}_{hslice}$. To that aim, we create a new box $\mathcal{B}_{disc}$ of width W and height $4\varepsilon OPT$ that we place on top of the current packing, hence increasing the total height to $(\frac{3}{2}+15\varepsilon)OPT$. Notice that each $R\in\mathcal{S}^{disc}_{hslice}$ satisfies $w(R)\le\varepsilon W$ and $h(R)=1\le\delta OPT$. Hence assuming $\varepsilon\le 1/4$ and $\delta\le 4\varepsilon^2$, Lemma 2 with $\varepsilon'=1/4$ guarantees that $\mathcal{S}^{disc}_{hslice}$ can be fully packed in $\mathcal{B}_{disc}$ via NFDH.

Properties (1) and (2) follow by contruction. It remains to prove (3). Notice that by construction the area inside $\mathcal{B}_{up}$ not occupied by $\mathcal{S}^{sel}_{hslice}$ is at most $\varepsilon OPT\cdot W$. Indeed, as observed earlier, if we take any point (x,y) along C_{up}, where x does not correspond to a step of C_{up}, the segment $x\times[y,y+\varepsilon OPT]$ hits some rectangle in $\mathcal{S}^{sel}_{hslice}$. Thus $a(\mathcal{B}_{up})\le a(\mathcal{S}^{sel}_{hslice})+\varepsilon OPT\cdot W$. A symmetric argument shows that $a(\mathcal{B}_{dw})\le a(\mathcal{T})+\varepsilon OPT\cdot W$. We can therefore conclude that

$$\begin{aligned}
a(\mathcal{B}_{free}) &\ge W\cdot\left(\frac{3}{2}+11\varepsilon\right)OPT - a(\mathcal{S}^{sel}_{hslice}) - a(\mathcal{T}) - 2\varepsilon OPT\cdot W\\
&\ge W\cdot\left(\frac{3}{2}+9\varepsilon\right)OPT - a(\mathcal{S}\cup\mathcal{T})\\
&\ge W\cdot\left(\frac{3}{2}+9\varepsilon\right)OPT - OPT\cdot W + a(\mathcal{V}) = W\cdot\left(\frac{1}{2}+9\varepsilon\right)OPT + a(\mathcal{V}).
\end{aligned}$$

◀

3.2 Including $\mathcal{V}_{vslice}$

In this section, we show how to incorporate $\mathcal{V}_{vslice}$ in the packing from the previous subsection. Critically, we need to leave a free box of sufficiently large size.

▶ **Lemma 7.** *Consider the packing from Lemma 6 and assume ε is small enough. It is possible to pack $\mathcal{V}_{vslice}$ inside the free boxes and furthermore define an empty rectangular region of size $2\varepsilon^2 W\times\frac{1}{2}OPT$.*

Proof. Consider the set of (at most) $2/\varepsilon$ free boxes $\mathcal{B}_1,\dots,\mathcal{B}_q$ sorted non-decreasingly by height. We partition them into unit-width vertical stripes $\mathcal{S}'=\{S_1,\dots,S_k\}$ sorted in the same order, and breaking ties so that stripes of the same box appear consecutively. Recall that $a(\mathcal{S}')\ge\left(\frac{1}{2}+9\varepsilon\right)OPT\cdot W+a(\mathcal{V})$.

We next place slices of $\mathcal{V}_{vslice}$ in these stripes from bottom to top in a greedy manner. In particular, we consider rectangles R in $\mathcal{V}_{vslice}$ in any order, and place R in the left-most stripe where it fits, as low as possible. Assume that the non-empty stripes are $\mathcal{S}'_{used}$. Notice that the unused space in these stripes is at most $(w(\mathcal{S}'_{used})-1)(\frac{OPT}{2}-1)+OPT\le\left(\frac{w(\mathcal{S}'_{used})+1}{2}\right)OPT$, hence

$$a(\mathcal{S}'_{used})\le a(\mathcal{V})+\frac{w(\mathcal{S}'_{used})+1}{2}OPT.$$

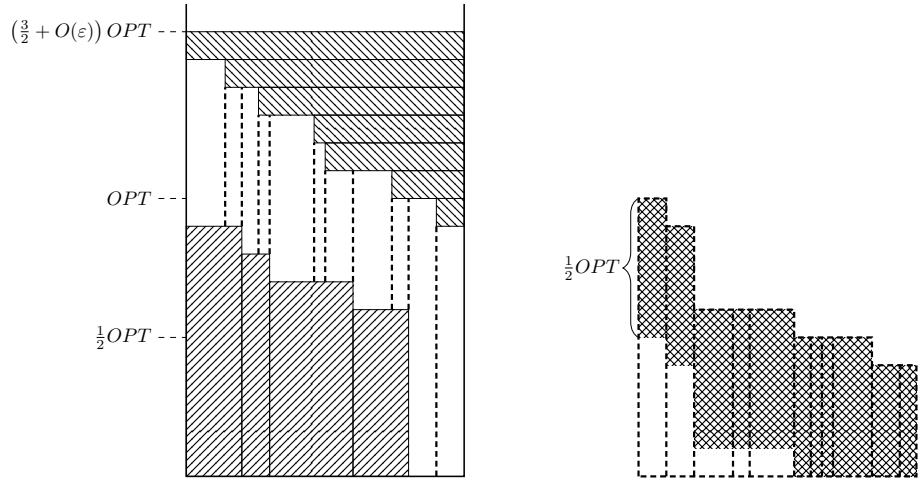


Figure 2 Description of the packing of $\mathcal{V}_{vslice}$. **Left:** Packing of $\mathcal{T} \cup \mathcal{S}_{hslice}$ as described in Lemma 6 and the $O_\varepsilon(1)$ boxes for the free area defined by the dashed lines. **Right:** Boxes in the free area sorted by height. Even if we ignore $\frac{1}{2}OPT$ height from each box we have enough space for $\mathcal{V}_{vslice}$ and even to reserve space for future discarded vertical rectangles.

Let us distinguish the unused stripes $\mathcal{S}'_{unused}$ into the ones $\mathcal{S}'^{tall}_{unused}$ of height at least $OPT/2$, and the remaining ones $\mathcal{S}'^{short}_{unused}$. One has that

$$\begin{aligned} a(\mathcal{S}'^{tall}_{unused}) &\geq \Big(\frac{1}{2}+9\varepsilon\Big)OPT\cdot W + a(\mathcal{V}) - a(\mathcal{S}'_{used}) - a(\mathcal{S}'^{short}_{unused}) \\ &\geq \Big(\frac{1}{2}+9\varepsilon\Big)OPT\cdot W + a(\mathcal{V}) - \Big(a(\mathcal{V}) + \frac{\big(w(\mathcal{S}'_{used})+1\big)}{2}OPT\Big) \\ &\quad - \frac{OPT}{2}w(\mathcal{S}'^{short}_{unused}) \\ &= \Big(\frac{1}{2}+9\varepsilon\Big)OPT\cdot W - \frac{\big(w(\mathcal{S}'_{used})+w(\mathcal{S}'^{short}_{unused})\big)}{2}OPT - \frac{OPT}{2} \\ &\geq 9\varepsilon OPT\cdot W - \frac{OPT}{2} \geq 8\varepsilon OPT\cdot W, \end{aligned}$$

The second last inequality follows from the fact that $\big(w(\mathcal{S}'_{used}) + w(\mathcal{S}'^{short}_{unused})\big) \leq W$. In the last inequality, we used the assumption $W \geq 1/\varepsilon$.

Since $a(\mathcal{S}'^{tall}_{unused}) \leq (\frac{3OPT}{2} + 11\varepsilon OPT)w(\mathcal{S}'^{tall}_{unused})$, it follows that, for $\varepsilon \leq 1/22$,

$$w(\mathcal{S}'^{tall}_{unused}) \geq \frac{8\varepsilon\cdot W}{\frac{3}{2}+11\varepsilon} \geq 4\varepsilon\cdot W.$$

Next, consider the set of boxes spanned by $\mathcal{S}'^{tall}_{unused}$. All these boxes contain a free rectangular region of height $\frac{1}{2}OPT$ induced by the bottom part of $\mathcal{S}'^{tall}_{unused}$: let us call these regions $F_1,\ldots,F_k$. Since the number of these regions is at most $\frac{2}{\varepsilon}$ (i.e. the total number of boxes) and their total width is at least $4\varepsilon\cdot W$, by an averaging argument there exists one such F_i of width at least $2\varepsilon^2 W$. ◀

3.3 Rounding

In this section, we show how to round the packing from Lemma 7 by means of Lemma 4, hence concluding the proof of Theorem 5.

Proof of Theorem 5. We start with the packing of $\mathcal{T} \cup \mathcal{S}_{hslice} \cup \mathcal{V}_{vslice}$ obtained from Lemma 7. Recall that $\mathcal{S}_{hslice}$ is packed into $1/\varepsilon + 1$ boxes $\mathcal{B}_S$ and $\mathcal{V}_{vslice}$ into $2/\varepsilon$ boxes $\mathcal{B}_V$. The total height of this packing is $(\frac{3}{2} + 15\varepsilon)OPT$, and this packing leaves a free region F of size $2\varepsilon^2 W \times \frac{1}{2}OPT$. Provided that δ is small enough, we can apply Lemma 4 to $(\mathcal{S}_{hslice}, \mathcal{B}_S)$ and obtain a packing of $\mathcal{S}$ into a set of containers fully contained in $\mathcal{B}_S$, plus two containers of size at most $W \times \frac{\varepsilon^2}{2}OPT$ each. We place the latter two containers on top of the packing, hence increasing the total height by $\varepsilon^2 OPT$. By applying the same Lemma to $(\mathcal{V}_{vslice}, \mathcal{B}_V)$, we obtain a packing of $\mathcal{V}$ into a set of containers fully contained in $\mathcal{B}_V$ plus two containers of size at most $\frac{\varepsilon^2}{2}W \times \frac{1}{2}OPT$ each. The latter two containers can be placed inside F without further increasing the height of the packing, still leaving a free region of size $\varepsilon^2 W \times \frac{1}{2}OPT$.

By construction, the number of used containers is $O_\varepsilon(1)$ and their sizes belong to a set that can be computed in polynomial time. ◀

4 Algorithm

In this section, we describe an algorithm based on Theorem 5 to compute our final solution.

Consider the set of containers guaranteed by Theorem 5. In polynomial time we can guess such containers by trying all possibilities. By brute force we can also compute (in polynomial time) a packing of these containers plus the free box in the strip of total height at most $(\frac{3}{2} + O(\varepsilon))OPT$. We guess which ones among the vertical containers contain $\mathcal{T}$, and pack the whole set $\mathcal{T}$ there greedily in non-increasing order of height.

We next apply Lemma 3 with parameter $\varepsilon' = \varepsilon^3$ to the remaining containers and to the remaining rectangles $\mathcal{V} \cup \mathcal{S}$. This way we can pack a set $\mathcal{R}' \subseteq \mathcal{V} \cup \mathcal{S}$ of area at least $a(\mathcal{V} \cup \mathcal{S})(1 - \varepsilon^3)$. It remains to pack $\mathcal{R}'' := (\mathcal{V} \cup \mathcal{S}) \setminus \mathcal{R}'$, $a(\mathcal{R}'') \leq \varepsilon^3 a(\mathcal{V} \cup \mathcal{S}) \leq \varepsilon^3(\frac{3}{2} + O(\varepsilon))OPT \cdot W \leq 2\varepsilon^3 OPT \cdot W$. We partition $\mathcal{R}''$ into 3 subsets and pack them as follows:

1. The rectangles $\mathcal{V}'' \subseteq \mathcal{R}''$ of height at least $2\varepsilon OPT$ (notice that they have height at most $OPT/2$). By an area argument their total width is at most $\frac{2\varepsilon^3 OPT \cdot W}{2\varepsilon OPT} = \varepsilon^2 W$. Hence they fit in a vertical container of size $\varepsilon^2 W \times \frac{1}{2}OPT$ that can be placed in the area occupied by the free box (without increasing the height of the packing).
2. The rectangles $\mathcal{H}'' \subseteq \mathcal{R}''$ of width at least $\varepsilon^2 W$. By a similar area argument their total height is at most $\frac{2\varepsilon^3 OPT \cdot W}{\varepsilon^2 W} = 2\varepsilon OPT$. Hence they can be placed into a horizontal container of size $W \times 2\varepsilon OPT$ that can be placed on top of the current packing.
3. The remaining rectangles $\mathcal{S}'' \subseteq \mathcal{R}''$ with height at most $2\varepsilon OPT$ and width at most $\varepsilon^2 W$. By Lemma 2 with parameter $\varepsilon' = \sqrt{\varepsilon}$ and for small enough ε, we can pack $\mathcal{S}''$ by means of NFDH into an area container of size $\varepsilon\sqrt{\varepsilon}W \times 2\sqrt{\varepsilon}OPT$ to be placed on top of the current packing.

We now have all the ingredients to prove our main theorem.

Proof of Theorem 1. Consider the above algorithm. Clearly it runs in polynomial time for any fixed parameter $\varepsilon > 0$. Furthermore, for δ small enough, it generates a feasible packing of all rectangles of total height at most $\left(\frac{3}{2} + O(\sqrt{\varepsilon})\right) OPT$. Considering the initial rounding of OPT by a factor $(1+\varepsilon)$, this gives a $\frac{3}{2} + O(\sqrt{\varepsilon})$ approximation. The claim then follows by choosing ε appropriately. ◀

5 Hardness of Approximation

In this section, we prove that the lower bound of $\frac{3}{2}$ on the approximability of Strip Packing still holds in the case of δ-skewed instances.

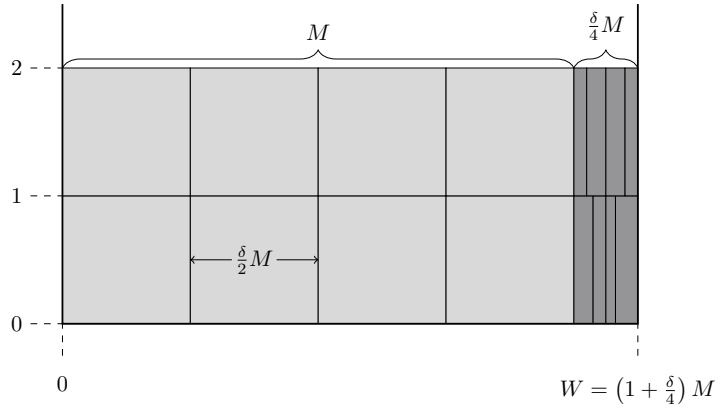


Figure 3 Construction from Lemma 8. Light gray rectangles represent dummy rectangles and dark gray rectangles represent partition rectangles.

▶ **Lemma 8.** *Given $\delta > 0$ and $\varepsilon > 0$, there is no polynomial-time $\left(\frac{3}{2} - \varepsilon\right)$-approximation for δ-skewed Strip Packing unless P=NP.*

Proof. We will prove this result via a reduction from the NP-complete *Partition* problem. Recall that in Partition we are given a set of integers $\mathcal{I} = \{x_1, \ldots, x_n\}$ whose sum is p. Our goal is to determine whether $\mathcal{I}$ can be partitioned into two sets $\mathcal{I}_1$ and $\mathcal{I}_2$ such that $\sum_{x_i \in \mathcal{I}_1} x_i = \frac{p}{2}$. We define our Strip Packing instance as follows: The width of the strip will be $W = (1 + \delta/4)M$ where $M = \frac{2p}{\delta}$. Also, we will have $n + \frac{4}{\delta}$ rectangles in the instance, from which $\frac{4}{\delta}$ will have height 1 and width $\frac{\delta}{2}M$ (*dummy rectangles*) and the remaining n rectangles will have, for each $i = 1, \ldots, n$, height 1 and width x_i (*partition rectangles*). Notice that the instance is indeed δ-skewed as the width of the rectangles is either $\frac{\delta}{2}M \leq \frac{\delta}{2}W$ or at most $p = \frac{\delta}{2}M \leq \frac{\delta}{2}W$. Notice also that $OPT \geq 2$ since the area of the rectangles is $2W$.

We will now prove that the Partition instance is a YES instance if and only if $OPT = 2$. Since all the heights in the instance are 1, as a consequence a NO instance has height at least 3, hence concluding the proof of the claim. Notice that if the Partition instance is a YES instance then we can pack one next to the other $\frac{2}{\delta}$ dummy rectangles plus one side of the partition since their total width would be $M + \frac{p}{2} = \left(1 + \frac{\delta}{4}\right) M$. We then analogously pack the rest of the rectangles on top, obtaining a packing of height 2 which is optimal as the total area of the rectangles is $2W$ (see Figure 3). On the other hand, if the optimal height of the Strip Packing instance is 2, the subregion $[0, W] \times [0, 2]$ in the strip must be fully occupied by rectangles. This actually implies that the horizontal segment $[0, W] \times \{1\}$ does not intersect the interior of any rectangle in the packing: indeed otherwise the space below that rectangle could not be occupied by any other rectangle (as heights are all 1). This divides the solution into two rows of height 1 and width W which are completely filled with rectangles. The only way to divide dummy rectangles into the two rows is to have exactly $\frac{2}{\delta}$ in each row (as the largest total width below W that they can sum up to is M and their total width is $2M$), hence the remaining partition rectangles in each row have total width exactly $\frac{p}{2}$, forming then a feasible solution to the Partition instance. ◀

References

1 Anna Adamaszek, Tomasz Kociumaka, Marcin Pilipczuk, and Michal Pilipczuk. Hardness of approximation for strip packing. *ACM Transactions on Computation Theory*, 9(3):14:1–14:7, 2017. doi:10.1145/3092026.

2 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 400–409. IEEE Computer Society, 2013. doi:10.1109/FOCS.2013.50.

3 Anna Adamaszek and Andreas Wiese. A quasi-ptas for the two-dimensional geometric knapsack problem. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1491–1505. SIAM, 2015. doi:10.1137/1.9781611973730.98.

4 Brenda S. Baker, Edward G. Coffman Jr., and Ronald L. Rivest. Orthogonal packings in two dimensions. *SIAM Journal on Computing*, 9(4):846–855, 1980. doi:10.1137/0209064.

5 Nikhil Bansal, Alberto Caprara, Klaus Jansen, Lars Prädel, and Maxim Sviridenko. A structural lemma in 2-dimensional packing, and its implications on approximability. In Yingfei Dong, Ding-Zhu Du, and Oscar H. Ibarra, editors, *Algorithms and Computation, 20th International Symposium, ISAAC 2009, Honolulu, Hawaii, USA, December 16-18, 2009. Proceedings*, volume 5878 of *Lecture Notes in Computer Science*, pages 77–86. Springer, 2009. doi:10.1007/978-3-642-10631-6_10.

6 Nikhil Bansal, José R. Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: Inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31(1):31–49, 2006. doi:10.1287/moor.1050.0168.

7 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. 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 13–25. SIAM, 2014. doi:10.1137/1.9781611973402.2.

8 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. URL: http://dl.acm.org/citation.cfm?id=1496770.1496867.

9 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

10 Henrik I. Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017. doi:10.1016/j.cosrev.2016.12.001.

11 Edward Grady Coffman and John L. Bruno. *Computer and job-shop scheduling theory*. Wiley New York, 1976.

12 Edward G. Coffman Jr., János Csirik, Gábor Galambos, Silvano Martello, and Daniele Vigo. *Bin Packing Approximation Algorithms: Survey and Classification*, pages 455–531. Springer New York, New York, NY, 2013. doi:10.1007/978-1-4419-7997-1_35.

13 Leah Epstein and Rob van Stee. This side up! *ACM Transactions on Algorithms*, 2(2):228–243, 2006. doi:10.1145/1150334.1150339.

14 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. In Chris Umans, editor, *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 260–271. IEEE Computer Society, 2017. doi:10.1109/FOCS.2017.32.

15 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via L-packings. *CoRR*, abs/1711.07710, 2017. arXiv:1711.07710.

16 Waldo Gálvez, Fabrizio Grandoni, Salvatore Ingala, and Arindam Khan. Improved pseudo-polynomial-time approximation for strip packing. In Akash Lal, S. Akshay, Saket Saurabh, and Sandeep Sen, editors, *36th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2016, December 13-15, 2016, Chennai, India*, volume 65 of *LIPIcs*, pages 9:1–9:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.FSTTCS.2016.9.

17 M. R. Garey and David S. Johnson. "strong" np-completeness results: Motivation, examples, and implications. *Journal of the ACM*, 25(3):499–508, 1978. doi:10.1145/322077.322090.

18 Rolf Harren, Klaus Jansen, Lars Prädel, and Rob van Stee. A $(5/3 + \epsilon)$-approximation for strip packing. *Computational Geometry*, 47(2):248–267, 2014. doi:10.1016/j.comgeo.2013.08.008.

19 Rolf Harren and Rob van Stee. Improved absolute approximation ratios for two-dimensional packing problems. 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 177–189. Springer, 2009. doi:10.1007/978-3-642-03685-9_14.

20 Sören Henning, Klaus Jansen, Malin Rau, and Lars Schmarje. Complexity and inapproximability results for parallel task scheduling and strip packing. *Theory of Computing Systems*, 64(1):120–140, 2020. doi:10.1007/s00224-019-09910-6.

21 Klaus Jansen and Lars Prädel. A new asymptotic approximation algorithm for 3-dimensional strip packing. In Viliam Geffert, Bart Preneel, Branislav Rovan, Julius Stuller, and A Min Tjoa, editors, *SOFSEM 2014: Theory and Practice of Computer Science - 40th International Conference on Current Trends in Theory and Practice of Computer Science, Nový Smokovec, Slovakia, January 26-29, 2014, Proceedings*, volume 8327 of *Lecture Notes in Computer Science*, pages 327–338. Springer, 2014. doi:10.1007/978-3-319-04298-5_29.

22 Klaus Jansen and Malin Rau. Closing the gap for pseudo-polynomial strip packing. In Michael A. Bender, Ola Svensson, and Grzegorz Herman, editors, *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 62:1–62:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019. doi:10.4230/LIPIcs.ESA.2019.62.

23 Klaus Jansen and Malin Rau. Improved approximation for two dimensional strip packing with polynomial bounded width. *Theoretical Computer Science*, 789:34–49, 2019. doi:10.1016/j.tcs.2019.04.002.

24 Klaus Jansen and Roberto Solis-Oba. New approximability results for 2-dimensional packing problems. In Ludek Kucera and Antonín Kucera, editors, *Mathematical Foundations of Computer Science 2007, 32nd International Symposium, MFCS 2007, Ceský Krumlov, Czech Republic, August 26-31, 2007, Proceedings*, volume 4708 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2007. doi:10.1007/978-3-540-74456-6_11.

25 Klaus Jansen and Rob van Stee. On strip packing with rotations. 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 755–761. ACM, 2005. doi:10.1145/1060590.1060702.

26 Klaus Jansen and Guochuan Zhang. Maximizing the total profit of rectangles packed into a rectangle. *Algorithmica*, 47(3):323–342, 2007. doi:10.1007/s00453-006-0194-5.

27 Edward G. Coffman Jr., M. R. Garey, David S. Johnson, and Robert Endre Tarjan. Performance bounds for level-oriented two-dimensional packing algorithms. *SIAM Journal on Computing*, 9(4):808–826, 1980. doi:10.1137/0209062.

28 Mohammad M. Karbasioun, Gennady Shaikhet, Evangelos Kranakis, and Ioannis Lambadaris. Power strip packing of malleable demands in smart grid. In *Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013*, pages 4261–4265. IEEE, 2013. doi:10.1109/ICC.2013.6655233.

29 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000. doi:10.1287/moor.25.4.645.12118.

30 Arindam Khan. *Approximation algorithms for multidimensional bin packing*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2016. URL: http://hdl.handle.net/1853/54371.

31 Joseph Y.-T. Leung, Tommy W. Tam, C. S. Wong, Gilbert H. Young, and Francis Y. L. Chin. Packing squares into a square. *Journal on Parallel and Distributed Computing*, 10(3):271–275, 1990. doi:10.1016/0743-7315(90)90019-L.

32 Flávio Keidi Miyazawa and Yoshiko Wakabayashi. Packing problems with orthogonal rotations. In Martin Farach-Colton, editor, *LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina, April 5-8, 2004, Proceedings*, volume 2976 of *Lecture Notes in Computer Science*, pages 359–368. Springer, 2004. doi:10.1007/978-3-540-24698-5_40.

33 Giorgi Nadiradze and Andreas Wiese. On approximating strip packing with a better ratio than 3/2. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1491–1510. SIAM, 2016. doi:10.1137/1.9781611974331.ch102.

34 Ingo Schiermeyer. Reverse-fit: A 2-optimal algorithm for packing rectangles. In Jan van Leeuwen, editor, *Algorithms - ESA '94, Second Annual European Symposium, Utrecht, The Netherlands, September 26-28, 1994, Proceedings*, volume 855 of *Lecture Notes in Computer Science*, pages 290–299. Springer, 1994. doi:10.1007/BFb0049416.

35 Daniel Dominic Sleator. A 2.5 times optimal algorithm for packing in two dimensions. *Information Processing Letters*, 10(1):37–40, 1980. doi:10.1016/0020-0190(80)90121-0.

36 A. Steinberg. A strip-packing algorithm with absolute performance bound 2. *SIAM Journal on Computing*, 26(2):401–409, 1997. doi:10.1137/S0097539793255801.

37 Shaojie Tang, Qiuyuan Huang, Xiang-Yang Li, and Dapeng Wu. Smoothing the energy consumption: Peak demand reduction in smart grid. In *Proceedings of the IEEE INFOCOM 2013, Turin, Italy, April 14-19, 2013*, pages 1133–1141. IEEE, 2013. doi:10.1109/INFCOM.2013.6566904.

A Omitted Proofs

Proof of Lemma 4. We prove the claim for $\mathcal{H}$, the case of $\mathcal{V}'$ being symmetric. For a proper parameter $\alpha > 0$ to be fixed later, we define a rectangle R (and its horizontal slices) to be *narrow* if $w(R) \leq \alpha^2 W$ and *wide* otherwise. We temporarily remove narrow rectangles, and pack the wide ones.

The first step in our construction is to round up the widths of the wide slices, while discarding a small-area subset of them. Let $\beta > 0$ be a parameter to be fixed later. Let us sort the wide slices $\mathcal{H}^{wide}_{hslice}$ in non-increasing order of width, and let us partition the obtained sequence into subsequences $\mathcal{H}_1, \ldots, \mathcal{H}_{1/\beta}$ of total height $\beta h(\mathcal{H}^{wide}_{hslice})$ each (excluding possibly the last group that can have smaller height). For a group i, we define w_i^{min} as the minimum width in $\mathcal{H}_i$. For each $i = 1, \ldots, 1/\beta - 1$, we define an injection between $\mathcal{H}_{i+1}$ and $\mathcal{H}_i$. Next, we delete slices $\mathcal{H}_1$. Let $\mathcal{H}_{wdisc1}$ denote the rectangles of which we removed at least one slice. Notice that all rectangles having some slice in $\mathcal{H}_1$ have all their slices in $\mathcal{H}_1$ excluding possibly one rectangle (which has part of its slices in $\mathcal{H}_1$). Observe that $h(\mathcal{H}^{wide}_{hslice}) \leq \frac{H}{\alpha}$ since otherwise $a(\mathcal{H}^{wide}_{hslice}) \geq \alpha W \cdot h(\mathcal{H})$ would be too large to fit into the region of size $W \times H$. Hence $h(\mathcal{H}_1) = \beta h(\mathcal{H}) \leq \frac{\beta}{\alpha} H$. It follows that $h(\mathcal{H}_{wdisc1}) \leq (\frac{\beta}{\alpha} + \delta)H$. For any fixed α, this quantity is at most $\frac{(\varepsilon')^2}{2} H$ if $\beta \leq \alpha(\varepsilon')^2/4$ and $\delta \leq (\varepsilon')^2/4$.

For $i = 1, \ldots, 1/\beta - 1$, we temporarily increase the width of each $H \in \mathcal{H}_{i+1}$ to w_i^{min}, hence getting an enlarged slice $\overline{H}$. Then, we move each such $\overline{H}$ into the region that was occupied by the slice $H' \in \mathcal{H}_i$ associated with H according to the above injection. Notice that this is possible since we removed $\mathcal{H}_1$ and since $w(\overline{H}) = w_i^{min} \leq w(H')$. Let $\overline{\mathcal{H}}$ be the final set of enlarged slices. Observe that the number of possible distinct widths in $\overline{\mathcal{H}}$ is $1/\beta - 1$.

Let us focus on a specific box $B \in \mathcal{B}$ of size $w(B) \times h(B)$, and let $\mathcal{H}'_{hslice}$ be the slices contained in B. Next, we partition B into unit height stripes. We shift slices in each stripe as left as possible, and permute them so that slices in $\overline{\mathcal{H}}$ appear to the left of each stripe. We call a *configuration* C of a stripe the sequence of (enlarged) widths $(w_1, \ldots, w_q)$ of its slices in $\overline{\mathcal{H}}$ from left to right. Notice that there are $1/\beta - 1$ possible enlarged widths, and each stripe can contain at most $1/\alpha^2$ wide slices. Hence the number of possible configurations is at most $n_{conf} = \sum_{i=0}^{1/\alpha^2} (1/\beta - 1)^i \leq 2(1/\beta - 1)^{1/\alpha^2} \leq (1/\beta)^{1/\alpha^2}$.

We reorder the stripes in $\mathcal{H}'_{hslice}$ vertically so that equal configurations appear consecutively from top to bottom, and stripes without narrow rectangles appear at the bottom. Suppose that the number of stripes in B with a given configuration $C = (w_1, \ldots, w_q)$ is $h(C)$, and $A(C)$ is the corresponding region. We initially cover $A(C)$ by creating q consecutive horizontal containers of height $h(C)$ and width $w_1, \ldots, w_q$ respectively. These containers altogether cover all the wide slices in B. The width of each container belongs to a set that can be computed in polynomial time (it is the width of some input rectangle). In order to enforce the same property for their heights, we round down the height of each such container to the largest multiple $h'(C)$ of $\frac{\delta}{\gamma}H$ not larger than $h(C)$, for some parameter $\gamma > 0$ to be fixed later. The number of these containers is $n_{wcont} \leq K n_{conf}$.

We next use the obtained horizontal containers to place most of the wide rectangles. We consider the containers in non-increasing order of width and the slices of wide rectangles in the same order, breaking ties so that slices of the same rectangle appear consecutively. We also create a dummy final container of sufficient width and of height large enough to accommodate the total height of the wide slices minus the total height of the containers. Now, we place back the slices into the containers following the previous order. Notice that all slices will fit. We discard each wide rectangle whose slices are contained in 2 containers (3 is not possible) and all the wide rectangles whose slices are contained in the dummy final container. Let $\mathcal{H}_{wdisc2}$ be the set of discarded rectangles. Their total height is

$$h(\mathcal{H}_{wdisc2}) \leq n_{wcont}\delta H + n_{wcont}\frac{\delta}{\gamma}H.$$

The above quantity is at most $\frac{(\varepsilon')^2}{2}H$ for any choice of ε', α, β, and γ, provided that $\delta \leq \frac{(\varepsilon')^2\gamma}{4n_{wcont}} \leq \frac{(\varepsilon')^2\gamma}{4K(1/\beta)^{(1/\alpha^2)}}$.

So we packed all the wide rectangles into horizontal containers except for the set $\mathcal{H}_{wdisc} = \mathcal{H}_{wdisc1} \cup \mathcal{H}_{wdisc2}$. The latter set has, by the above discussion, height at most $(\varepsilon')^2 H$, hence we can pack it into a container of size $\max_{R \in \mathcal{H}^{wide}}\{w(R)\} \times (\varepsilon')^2 H$.

It remains to pack the narrow rectangles. Consider again a given box B. For each configuration C, there is some free region $F(C)$ to the right of the containers built for C whose height is $h'(C)$ (in particular, a multiple of $\frac{\delta}{\gamma}H$) and of some width $w(F(C))$. We build an area container of the same height and with width equal to the largest multiple $w'(F(C))$ of $\frac{\alpha}{\gamma}W$ not larger than $w(F(C))$. We apply a similar construction to the free rectangular region F in B below all the previous containers, if any: in particular we create an area container whose width is the largest multiple of $\frac{\alpha}{\gamma}W$ not larger than $w(F) = w$ and whose height is the largest multiple of $\frac{\delta}{\gamma}H$ not larger than $h(F)$. The total number of constructed area containers is $n_{ncont} \leq K \cdot n_{conf}$.

Next, we start packing the narrow rectangles in non-increasing order of height in the area containers using NFDH. Observe that these rectangles satisfy the claim of Lemma 2 with parameter γ. If all narrow rectangles are packed this way, we are done. Otherwise, let $\mathcal{H}_{ncont}$ and $\mathcal{H}_{ndisc}$ be the subset of narrow rectangles that are packed and not packed in the area containers, resp. By Lemma 2, $a(\mathcal{H}_{ncont}) \geq (1 - 2\gamma)A_{acont}$, where A_{acont} is the

total area of the area containers. Let a_{free} be the total area in the boxes not occupied by horizontal containers. Clearly $a_{free} \geq a(\mathcal{H}^{narrow})$ since all narrow slices did fit in a region of area not smaller than a_{free}. Due to the rounding involved in the construction, in each box there is some area which is not used by area containers nor by horizontal ones. The latter area is at most $W \cdot \frac{\delta}{\gamma}H + H \cdot \frac{\alpha}{\gamma}W$ per container, hence at most $\Delta \leq K \cdot (\frac{\delta+\alpha}{\gamma})WH$ in total. We can conclude that

$$a(\mathcal{H}_{ncont}) \geq (1-2\gamma)A_{acont} = (1-2\gamma)(a_{free}-\Delta) \geq (1-2\gamma)(a(\mathcal{H}^{narrow})-K\cdot(\frac{\delta+\alpha}{\gamma})WH).$$

Thus

$$a(\mathcal{H}_{ndisc}) \leq 2\gamma \cdot a(\mathcal{H}^{narrow}) + K \cdot (\frac{\delta+\alpha}{\gamma})WH) \leq (2\gamma + K \cdot (\frac{\delta+\alpha}{\gamma}))WH.$$

If we choose $\gamma \leq (\varepsilon')^2/6$, $\delta \leq \frac{(\varepsilon')^2\gamma}{6K}$ and $\alpha \leq \frac{(\varepsilon')^2\gamma}{6K}$, then the latter quantity is at most $\frac{(\varepsilon')^2}{2}WH$. Next, we create a new area container C_{darea} of size $\varepsilon' W \times \varepsilon' H$, and use NFDH to pack $\mathcal{H}_{ndisc}$ in it. It is not difficult to verify that, for such values of δ and α, rectangles in $\mathcal{H}_{ndisc}$ satisfy the conditions of Lemma 2 with parameter ε'. Thus we have

$$a(\mathcal{H}_{ndisc}) \leq \frac{1}{2}(\varepsilon')^2 WH = \frac{1}{2}a(C_{darea}) \leq (1-2\varepsilon')a(C_{darea}).$$

Thus all rectangles in $\mathcal{H}_{ndisc}$ are packed into C_{darea}.

It is possible to choose constant parameters α, β and γ such that the above conditions are all satisfied (for δ small enough) and the total number of containers is $O_{\varepsilon'}(1)$. More precisely, this is true if $\gamma = (\varepsilon')^2/6$, $\alpha = (\varepsilon')^4/(36 \cdot K)$, $\beta = (\varepsilon')^6/(144 \cdot K)$ and $\delta = \frac{(\varepsilon')^2\gamma}{4K(1/\beta)^{(1/\alpha^2)}} \in \delta \leq (\varepsilon'/K)^{(K/\varepsilon')^{O(1)}}$, leading to at most $(K/\varepsilon')^{(K/\varepsilon')^{O(1)}}$ containers. By the above construction, the sizes of the containers belong to a set that can be computed in polynomial time. ◀

Learning Lines with Ordinal Constraints

Bohan Fan
Department of Computer Science, University of Illinois at Chicago, IL, USA
bfan4@uic.edu

Diego Ihara
Department of Computer Science, University of Illinois at Chicago, IL, USA
https://dihara2.people.uic.edu/
dihara2@uic.edu

Neshat Mohammadi
Department of Computer Science, University of Illinois at Chicago, IL, USA
nmoham24@uic.edu

Francesco Sgherzi
Department of Computer Science, University of Illinois at Chicago, IL, USA
https://fsgher2.people.uic.edu/
fsgher2@uic.edu

Anastasios Sidiropoulos
Department of Computer Science, University of Illinois at Chicago, IL, USA
https://sidiropo.people.uic.edu/
sidiropo@uic.edu

Mina Valizadeh
Department of Computer Science, University of Illinois at Chicago, IL, USA
mvaliz2@uic.edu

Abstract

We study the problem of finding a mapping f from a set of points into the real line, under ordinal triple constraints. An ordinal constraint for a triple of points (u, v, w) asserts that $|f(u) - f(v)| < |f(u) - f(w)|$. We present an approximation algorithm for the dense case of this problem. Given an instance that admits a solution that satisfies $(1-\varepsilon)$-fraction of all constraints, our algorithm computes a solution that satisfies $(1 - O(\varepsilon^{1/8}))$-fraction of all constraints, in time $O(n^7) + (1/\varepsilon)^{O(1/\varepsilon^{1/8})}n$.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases metric learning, embedding into the line, ordinal constraints, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.45

Category APPROX

Related Version https://arxiv.org/abs/2004.13202

Supplementary Material https://github.com/lnghrdntcr/lloc

Funding Supported by NSF grants CCF-1815145, CCF-1934915, and by NSF CAREER award 1453472.

1 Introduction

Geometric methods provide several tools for the analysis of complicated data sets, such as nearest-neighbor search, clustering, and dimensionality reduction. The key abstraction is to encode a set of objects by mapping each object to a point in some metric space, such that the distance between points quantifies the pairwise dissimilarity between the corresponding

© Bohan Fan, Diego Ihara, Neshat Mohammadi, Francesco Sgherzi, Anastasios Sidiropoulos, and Mina Valizadeh;
licensed under Creative Commons License CC-BY

objects. The success of this paradigm crucially depends on the metrical representation used to encode the data. Motivated by this fact, metric learning aims at developing methods for discovering an underlying metric space from proximity information (we refer the reader to [23, 17] for a detailed exposition). There are several different formulations of the metric learning problem that have been considered in the literature. Here, we focus on the popular case of *ordinal* constraints. In this case, the input consists of a set of points $X = [n]$, together with a set $\mathcal{T}$ of ordered triples (u, v, w) of points, representing the fact that u is *more similar* to v than to w. The goal is to find a mapping $f : X \to Y$, for some host metric space (Y, ρ), such that for all $(u, v, w) \in \mathcal{T}$, we have

$$\rho(f(u), f(v)) < \rho(f(u), f(w)). \tag{1}$$

In general, there might be no mapping f that satisfies all constraints of the form (1), so we are interested in the algorithmic problem of computing a mapping that minimizes the fraction of violated constraints. We focus on the case where the host space is the real line, so the objective can be formulated as computing a mapping $f : [n] \to \mathbb{R}$, where for each $(u, v, w) \in \mathcal{T}$ we have the constraint

$$|f(u) - f(v)| < |f(u) - f(w)|. \tag{2}$$

We refer to this problem as Line Learning with Ordinal Constraints (LLOC).

1.1 Our contribution

We present an approximation algorithm for learning a line metric space under ordinal constraints, for the case of dense instances. Here, the density condition means that all ordinal information is given, i.e. for any distinct $u, v, w \in [n]$, we have either $(u, v, w) \in \mathcal{T}$, or $(u, w, v) \in \mathcal{T}$. Our main result is summarized in the following.

▶ **Theorem 1.1.** *There exists an algorithm that given an instance of LLOC that admits a solution satisfying $(1 - \varepsilon)$-fraction of all constraints, outputs a solution that satisfies $(1 - O(\varepsilon^{1/8}))$-fraction of all constraints, in time $O(n^7) + (1/\varepsilon)^{O(1/\varepsilon^{1/8})} n$.*

Brief overview of our approach

The main idea used to obtain Theorem 1.1 is to first compute an ordering that is close to the ordering of the points in the optimal solution. This is done by "guessing" a point p^* that lies within the few left-most points in an optimal solution, and such that p^* is not involved in many violated constraints. We show that the ordinal constraints involving p^* can be used to order the points by first solving an instance of the Minimum Feedback Arc Set problem on a tournament, and then computing a topological ordering of the remaining acyclic graph. We use this ordering to partition the points into "buckets", and we show that for almost all buckets, almost all their points must be mapped inside an interval that does not contain many other points. This property allows us to define a smaller instance of the problem by contracting each bucket into a single point. This new smaller instance can be solved exactly, and its solution can be pulled back to the original problem.

1.2 Related work

Metric learning

Another popular formulation of the metric learning problem uses *contrastive* constraints. In this case, the input consists of a set of points $X = [n]$, together with sets $\mathcal{S}, \mathcal{D} \subseteq \binom{X}{2}$, where $\mathcal{S}$ contains pairs labeled as *similar*, and $\mathcal{D}$ contains pairs labeled as *dissimilar*. The goal is to find a mapping $f : X \to Y$, for some host metric space (Y, ρ), such that for all $\{u, v\} \in \mathcal{S}$,

$$\rho(f(u), f(v)) \leq \ell,$$

and for all $\{u, v\} \in \mathcal{D}$,

$$\rho(f(u), f(v)) \geq h,$$

for some given threshold values $\ell, h > 0$. This problem is easily seen to be a generalization of Correlation Clustering. It has been studied for the case dense instances, when the host metric space is either Euclidean or a tree [13]. The main result of [13] is a FPTAS for the case where there exists a mapping that satisfies all constraints, that is allowed to violate the constraints by a small multiplicative factor which is referred to as *contrastive distortion*. In contrast, in the present work, we do not introduce any distortion, and we do not need to assume that there exists a mapping satisfying all the constraints.

We also note that the case of arbitrary instances (i.e., not necessarily dense) under contrastive constraints has been studied for the setting of learning Mahalanobis metric spaces (i.e., when X is a set of points in d-dimensional Euclidean space, and f is required to be linear) [12]. This version of the problem is related to the theory of LP-type problems.

Embedding into the line

The problem of computing a geometric representation of a data set into the real line has been studied extensively in various forms. This is arguably the simplest instance of dimensionality reduction, which is also a prototypical unsupervised metric learning task. Various objectives have been studied, including multiplicative [19, 20, 6, 5, 7, 11], additive [3], and average [10, 22] distortion. We refer the read to [14] for a detailed exposition. A related notion is ordinal embeddings, where one seeks to obtain mappings that approximately preserve the relative ordering of pairwise distances [2, 4]. We remark that a key difference between these works and our result is that they seek to minimize the *ordinal distortion*, which is a multiplicative factor of violation of the ordinal constraints, while we are interested in minimizing the number of violated ordinal constraints (without introducing ordinal distortion).

Betweenness

In the Betweenness problem we are given some set $X = [n]$ and a set $\mathcal{T}$ of ordered triples $(a, b, c) \in [n]^3$. The goal is to find a bijection $g : [n] \to [n]$ such that for any $(a, b, c) \in \mathcal{T}$, $g(b)$ appears between $g(a)$ and $g(c)$. This problem has been studied extensively in the literature. It is known to be MAXSNP-hard [9] (see also [21]), and remains hard to approximate even on dense instances [1]. The case of tournaments has been shown to admit a PTAS [15], while the best approximation algorithm for general instances is the 1/3-approximation obtained by taking a uniformly random ordering, assuming the Unique Games conjecture [8] (see also [18]).

The Betweenness problem is conceptually similar to the Line Learning with Ordinal Constraints problem studied here. However, as we now explain, the two problems have some important differences. A first difference is that the ordinal constraint (2) does not imply any

ordering constraint[1]. A second difference is that the solution space to the Line Learning with Ordinal Constraints problem that we study is larger. In other words, the ordering of the points is not always enough to recover a nearly-optimal constraint. For example, consider the instance on $X = \{0, 2, 4, \ldots, 2k, 2k+1, \ldots, 3k\}$, with all constraints $(u, v, w) \in X^3$, such that $|u - v| < |u - w|$. Clearly, setting f to be the identity results in a solution that satisfies all constraints. However, just the ordering of the points in f is not enough to obtain a good solution: setting $g(u_i) = i$, where $g(u_1) < g(u_2) < \ldots < g(u_n)$ results in a solution g that violates a constant fraction of all constraints.

1.3 Organization

The rest of the paper is organized as follows. Section 2 presents, as a warm up, an exact polynomial-time algorithm for the case where there exists a solution that satisfies all constraints. Section 3 presents the algorithm for the general case. Section 4 presents the analysis. Section 5 gives the proof of a technical Lemma which is used in the proof of the main result.

2 Warm up: An exact algorithm with no violations

We now describe an exact polynomial-time algorithm for the case where there exists an optimal solution that satisfies all constraints. This algorithm is significantly simpler than the one used to prove our main result. However, it illustrates the main idea of using the constraints involving some point p to deduce an ordering of all points, and then using this ordering to obtain an embedding into the line. The algorithm is summarized in the following.

▶ **Theorem 2.1.** *Let ε^* be the fraction of violated constraints, then there exists a polynomial-time algorithm which given an instance $([n], \mathcal{T})$ of the LLOC problem, either computes a mapping $f : [n] \to \mathbb{R}$ that satisfies all the constraints, or correctly decides that no such mapping exists.*

Proof. Fix some optimal mapping $f^* : [n] \to \mathbb{R}$, that satisfies all constraints in $\mathcal{T}$. We guess $p = \arg\min_{x \in [n]} f^*(x)$. For all $i, j \in [n]$, let $d_{i,j} = |f^*(x_j) - f^*(x_i)|$. We first determine the ordering of all the points on the real line, and then we compute the mapping using their distance constraints and solving some LP.

Suppose that $[n] = \{x_1, \ldots, x_n\}$, such that

$$f^*(p) = f^*(x_1) < f^*(x_2) < \ldots < f^*(x_n).$$

Since $\varepsilon^* = 0$, it follows that for all $i < j \in [n]$, we have $d_{1,i} < d_{1,j}$, and $(1, i, j) \in \mathcal{T}$. Therefore, for any $q, q' \in [n]$, we can decide whether $f^*(q) < f^*(q')$ or $f^*(q') < f^*(q)$ based on whether $(p, q, q') \in \mathcal{T}$ or $(p, q', q) \in \mathcal{T}$. Therefore, we can compute the ordering $x_1, \ldots, x_n$ of $[n]$ by running a sorting algorithm using pairwise comparisons.

We now compute a mapping using an LP. For any $i < j \in \{1, \ldots, n\}$, we have $|f^*(x_i) - f^*(x_j)| = \sum_{t=i}^{j-1} d_{t,t+1}$. Therefore for each $(x_i, x_j, x_k) \in \mathcal{T}$, the constraint $|f^*(x_i) - f^*(x_j)| < |f^*(x_i) - f^*(x_k)|$ can be written as $\sum_{t=i}^{j-1} d_{t,t+1} < \sum_{t=i}^{k-1} d_{t,t+1}$. Thus computing the desired mapping f can be done by computing a feasible solution to the following LP:

[1] For example, the constraint (u, v, w) is satisfied by both solutions $f(u) = 1, f(v) = 2, f(w) = 3$, and $f(u) = 2, f(v) = 1, f(w) = 4$, however the former solution implies the ordering $f(u) < f(v) < f(w)$, while the latter implies $f(v) < f(u) < f(w)$

$$d_{i,j} \geq 0 \text{ for all } i < j \in [n]$$
$$\sum_{t=i}^{j-1} d_{t,t+1} < \sum_{t=i}^{k-1} d_{t,t+1} \text{ for all } (x_i, x_j, x_k) \in \mathcal{T}$$

This concludes the proof. ◀

3 The algorithm for the general case

In this Section we present the algorithm for the general case of the problem. The algorithm uses as a subroutine an exact algorithm for a generalized weighted version of the problem. This exact algorithm is used on small instances that are constructed via a process which we refer to as a *retraction*.

3.1 Retractions

We now define a weighted version of the metric learning problem, where each constraint is associated with some weight, and the goal is to maximize the total weight of all satisfied constraints. Formally, an input to the Weighted Line Learning with Ordinal Constraints (WLLOC) problem is defined by a tuple $([b], \mathcal{T}, w)$, where $b \in \mathbb{N}$, and $\mathcal{T}$ are as before, and $w : \mathcal{T} \to \mathbb{R}$ is a weight function. The goal is to find a solution $f : [b] \to [0, 1]$ that minimizes the total weight of violated constraints.

▶ **Theorem 3.1.** *There exists an exact algorithm for the WLLOC problem with running time* $O(n^{3n})$.

Proof. We identify the space of possible solutions with $[0, 1]^n$, by mapping each solution $f : [b] \to [0, 1]$ to the vector $x_f = (f(1), \ldots, f(n)) \in [0, 1]^n$. For any $(i, j, k) \in \mathcal{T}$, we have the constraint

$$|f(i) - f(j)| < |f(i) - f(k)|.$$

The feasible region for this constraint is thus defined as a union of certain cells in an arrangement $A_{(i,j,k)}$ of a constant number of open halfspaces in $\mathbb{R}^n$. Let A be the arrangement obtained as the union of all halfspaces for all $(i, j, k) \in \mathcal{T}$. It is known that any arrangement of a halfspaces in $\mathbb{R}^b$ has complexity $O(a^b)$ (see [24] and references therein), and thus A has complexity $O(|\mathcal{T}|^n) = O(n^{3n})$. By enumerating all the cells in this arrangement, we find a solution that satisfies a set of constraints of maximum total weight, which results in an algorithm with running time $O(n^{3n})$. ◀

As mentioned earlier, the exact algorithm from Theorem 3.1 will be used as a subroutine on smaller instances. The following Definition describes a process for mapping large unweighted instances to smaller weighted ones.

▶ **Definition 3.2** (Retraction). *Given an instance* $\phi = ([n], \mathcal{T})$ *of the LLOC problem, and some partition* $\mathcal{B} = \{B_1, \ldots, B_b\}$ *of* $[n]$*, we define the* $\mathcal{B}$*-retraction of* ϕ *to be the instance* $\phi' = ([b], \mathcal{T}', w)$ *of the WLLOC problem where for any* $(i, j, k) \in \mathcal{T}'$*, we have*

$$w((i, j, k)) = |\mathcal{T} \cap (B_i \times B_k \times B_j)|.$$

3.2 The algorithm

The last ingredient we need is an approximation algorithm for the Minimum Feedback Arc Set problem on tournaments, which is summarized in the following.

▶ **Theorem 3.3** (Kenyon-Mathieu & Schudy [16]). *There exists a randomized algorithm for the Minimum Feedback Arc Set problem on weighted tournaments. Given $\epsilon > 0$, it outputs a solution with expected cost at most $(1+\epsilon)\mathsf{OPT}$. The expected running time is $O(1/\varepsilon)n^6 + 2^{\tilde{O}(1/\varepsilon)}n^2 + 2^{2^{\tilde{O}(1/\varepsilon)}}n$.*

We are now ready to describe the general algorithm. Let $\mathcal{T}_n$ denote the set of all ordered triples of distinct elements in $[n]$. Recall that the input consists of a set $\mathcal{T} \subseteq \mathcal{T}_n$, such that for any set of distinct $i, j, k \in [n]$, we have that exactly one of the triples (i, j, k) and (i, k, j) is contained in $\mathcal{T}$.

The algorithm proceeds in the following steps:

Step 1: Exhaustively computing a left-most point. Iterate Steps 2–5 for all values $p \in [n]$.

Step 2: Cycle removal. Construct a tournament $G^{(p)} = ([n], A^{(p)})$, where

$$A^{(p)} = \{(i, j) : (p, i, j) \in \mathcal{T}\}.$$

Compute an $O(1)$-approximate minimum feedback arc set, $F^{(p)} \subset A^{(p)}$, in $G^{(p)}$, using the algorithm in Theorem 3.3.

Step 3: Ordering. Compute a topological ordering $z_1^{(p)}, \ldots, z_n^{(p)}$ of $G^{(p)} \setminus F^{(p)}$.

Step 4: Retraction. Let $b = O(\varepsilon^{-1/8})$. For any $i \in [b]$, let

$$\mathcal{B}_i^{(p)} = \bigcup_{j=(i-1)n/b+1}^{in/b} \{z_j^{(p)}\}.$$

Let $\psi^{(p)}$ be the $\mathcal{B}^{(p)}$-retraction of $\phi^{(p)}$.

Step 5: Extension. Using the algorithm from Theorem 3.1, we compute an optimal solution $g^{(p)} : [b] \to [0, 1]$ for the instance $\psi^{(p)}$ of WLLOC. We define $f^{(p)} : [n] \to [0, 1]$ by setting for any $i \in [n]$, $f^{(p)}(i) = g^{(p)}(j)$, where $j \in [b]$ such that $i \in B_j^{(p)}$. The algorithm outputs the solution $f^{(p)}$.

Step 6: Return the best solution found among $f^{(1)}, \ldots, f^{(n)}$.

This completes the description of the algorithm.

4 Analysis of the algorithm

This Section presents the analysis of the algorithm, which is the proof of Theorem 1.1.

For the remainder of the analysis, let us fix some optimal solution $f_{\mathsf{OPT}} : [n] \to [0, 1]$ for the instance $([n], \mathcal{T})$ of the LLOC problem. Fix a numbering $\{x_1, \ldots, x_n\} = [n]$, such that

$$f_{\mathsf{OPT}}(x_1) \leq f_{\mathsf{OPT}}(x_2) \leq \ldots \leq f_{\mathsf{OPT}}(x_n).$$

For any $f : [n] \to [0, 1]$, for any $i \in [n]$, and for any $\alpha \in [0, 1]$, we say that i is *α-good in f*, if at least α-fraction of the constraints of the form $(i, j, k) \in \mathcal{T}$ are satisfied; i.e.:

$$|\{(i, j, k) \in \mathcal{T} : |f(i) - f(j)| < |f(i) - f(k)|\}| \geq \alpha \binom{n-1}{2}.$$

We first argue that there exists a $(1 - \varepsilon^{1/2})$-good point that is close to the left-most point in the optimal solution:

▶ **Lemma 4.1.** *There exists $i^* \in [2\varepsilon^{1/2}n]$, such that x_{i^*} is $(1-\varepsilon^{1/2})$-good in f_{OPT}.*

Proof. Let ξ be the total number of constraints violated by f_{OPT}. We have $\xi \leq \varepsilon \cdot |\mathcal{T}| = \varepsilon n\binom{n-1}{2}$. Suppose that there exists no $i \in [2\varepsilon^{1/2}n]$ such that x_i is $(1-\varepsilon^{1/2})$-good. Therefore every $i \in [2\varepsilon^{1/2}n]$ participates in at least $\varepsilon^{1/2}\binom{n-1}{2}$ violated constraints of the form (i,j,k), for some $j,k \in [n]$. Thus the total number of violated constraints is at least $\xi \geq 2n\varepsilon\binom{n-1}{2}$, which is a contradiction, concluding the proof. ◀

For the remainder of this section, fix some $i^* \in [2\varepsilon^{1/2}n]$, such that x_{i^*} is $(1-\varepsilon^{1/2})$-good, as in Lemma 4.1. Let f' be the embedding obtained from f_{OPT} by exchanging the images of x_1 and x_{i^*}, that is for all $i \in [n]$,

$$f'(x_i) = \begin{cases} f_{\mathsf{OPT}}(x_{i^*}) & \text{if } i = 1 \\ f_{\mathsf{OPT}}(x_1) & \text{if } i = i^* \\ f_{\mathsf{OPT}}(x_i) & \text{otherwise} \end{cases}$$

We next show that f' is near-optimal.

▶ **Lemma 4.2.** *The total number of violated constraints in f' is at most $(\varepsilon + O(1/n))n\binom{n-1}{2}$.*

Proof. Let $\mathcal{T}_1 \subseteq \mathcal{T}$ be the set of constraints that are violated in f' and in f_{OPT}. Let $\mathcal{T}_2 \subseteq \mathcal{T}$ be the set of constraints that are violated in f' but not in f_{OPT}. We have $|\mathcal{T}_1| \leq \varepsilon n\binom{n-1}{2}$. Since f_{OPT} and f' differ only on x_1 and x_{i^*}, it follows that every constraint $(i,j,k) \in \mathcal{T}_2$ must contain at least one of 1 and i^*. There are at most $6n^2$ such constraints. Thus $|\mathcal{T}_2| \leq 6n^2$. We conclude that the total number of constraints violated in f' is at most $|\mathcal{T}_1| + |\mathcal{T}_2| \leq (\varepsilon + O(1/n))n\binom{n-1}{2}$, which concludes the proof. ◀

The next Lemma shows that x_{i^*} remains $(1-O(\varepsilon^{1/2}))$-good in f'.

▶ **Lemma 4.3.** *We have that x_{i^*} is $(1-O(\varepsilon^{1/2}))$-good in f'.*

Proof. Let $\gamma = (x_{i^*}, j, k) \in \mathcal{T}$, and suppose that γ is satisfied in f_{OPT}. If

$$f_{\mathsf{OPT}}(x_{i^*}) \leq f_{\mathsf{OPT}}(j) \leq f_{\mathsf{OPT}}(k),$$

then, since $f'(j) = f_{\mathsf{OPT}}(j)$, and $f'(k) = f_{\mathsf{OPT}}(k)$, it follows that

$$f'(x_{i^*}) \leq f'(j) \leq f'(k),$$

and thus γ is also satisfied in f'.

Thus, the only possible constraints of the form $(x_{i^*}, j, k) \in \mathcal{T}$, that are not violated in f_{OPT}, but are violated in f', must satisfy either $f_{\mathsf{OPT}}(j) < f_{\mathsf{OPT}}(x_{i^*})$, or $f_{\mathsf{OPT}}(k) < f_{\mathsf{OPT}}(x_{i^*})$. In other words, we must have $\{j,k\} \cap \{x_1, \ldots, x_{i^*-1}\} \neq \emptyset$. Therefore, there are at most $2\varepsilon^{1/2}n^2$ such constraints. Since x_{i^*} is $(1-\varepsilon^{1/2})$-good in f_{OPT}, it follows that x_{i^*} is $(1-O(\varepsilon^{1/2}))$-good in f', which concludes the proof. ◀

Let $F' = \{(j,k) \in A^{(i^*)} : (x_{i^*}, j, k) \in \mathcal{T}$ and f' violates $(x_{i^*}, j, k)\}$. The next Lemma shows F' is a valid feedback arc set for $G^{(i^*)}$.

▶ **Lemma 4.4.** *F' is a feedback arc set for $G^{(i^*)}$, with $|F'| \leq (O(\varepsilon^{1/2}))\binom{n-1}{2}$.*

Proof. By Lemma 4.3, x_{i^*} is $(1 - O(\varepsilon^{1/2}))$-good, and thus $|F'| \leq (O(\varepsilon^{1/2}))\binom{n-1}{2}$. Thus, it suffices to show that F' is a feedback vertex set. For any $(j,k) \in A^{(i^*)} \setminus F'$, we have that (x_{i^*}, j, k) is satisfied in f'. Since x_{i^*} is mapped to the left-most point in f', it follows that $f'(j) < f'(k)$. Thus,

$$x_{i^*}, x_2, x_3, \ldots, x_{i^*-1}, x_1, x_{i^*+1}, x_{i^*+2}, \ldots, x_n$$

is a topological ordering of $G^{(i^*)} \setminus F'$, and thus F' is a feedback arc set, which concludes the proof. ◀

If the instance admits a solution with no violations, then it can be shown that the bucketing $\mathcal{B}^{(i^*)}$ computed by the algorithm agrees with a partition of the optimal solution to contiguous disjoint intervals. In the following, we show that, in the general case, the bucketing is "close" to such a partition. First, we introduce a notion of "stability" which formalizes what it means for a bucket to be close to an optimal interval.

▶ **Definition 4.5** (Stability). *Let $i \in [b]$. We say that i is* stable *if there exists some interval $I \subset \mathbb{R}$, such that*

$$\left| I \cap f'\left(B_i^{(i^*)}\right)\right| \geq (1 - \varepsilon^{1/8}) \cdot n/b,$$

and

$$\left| I \cap f'\left([n] \setminus B_i^{(i^*)}\right)\right| \leq \varepsilon^{1/8} \cdot n/b,$$

We also say that i is I-stable. We say that i is unstable *(I-unstable) if it is not stable (I-stable).*

The following Lemma gives a characterization of unstable buckets.

▶ **Lemma 4.6.** *Suppose that $i \in [b]$ is unstable. Then there exist pairwise disjoint intervals $I_1, I_2, I_3 \subset \mathbb{R}$, that appear in this order from left to right in the line, such that*

$$\left| I_1 \cap f'\left(B_i^{(i^*)}\right)\right| \geq n\varepsilon^{1/8}/(2b),$$

$$\left| I_3 \cap f'\left(B_i^{(i^*)}\right)\right| \geq n\varepsilon^{1/8}/(2b),$$

and

$$\left| I_2 \cap f'\left([n] \setminus B_i^{(i^*)}\right)\right| > \varepsilon^{1/8} \cdot n/b,$$

Proof. Let $I_1 \subset \mathbb{R}$ be the minimal interval that contains the $n\varepsilon^{1/8}/(2b)$ left-most points in $f'(B_i^{(i^*)})$, and let $I_3 \subset \mathbb{R}$ be the minimal interval that contains the $n\varepsilon^{1/8}/(2b)$ right-most points in $f'(B_i^{(i^*)})$. Let $I_2 \subset \mathbb{R}$ be the maximal interval that is contained between I_1 and I_3. Since $\varepsilon^{1/8} < 1$, we have that $I_1 \cap I_3 = \emptyset$, and therefore, all intervals I_1, I_2, I_3 are well-defined and pairwise disjoint. By construction, I_1 and I_3 each contains exactly $n\varepsilon^{1/8}/(2b)$ points in $f'(B_i^{(i^*)})$. Therefore, it remains to show that I_2 contains more than $\varepsilon^{1/8}n/b$ points in $f'([n] \setminus B_i^{(i^*)})$. Suppose, for the sake of contradiction, that I_2 contains at most $\varepsilon^{1/8}n/b$ in $f'([n] \setminus B_i^{(i^*)})$. Then, I_2 contains exactly $(1 - \varepsilon^{1/8})n/b$ points in $f'(B_i^{(i^*)})$, and at most $\varepsilon^{1/8}n/b$ points in $f'([n] \setminus B_i^{(i^*)})$, implying that B_i is stable, which is a contradiction. This concludes the proof. ◀

We next show that for each unstable bucket, the feedback arc set must contain many edges incident to vertices in the bucket.

▶ **Lemma 4.7.** *Let $i \in [b]$ be unstable. Then, $F^{(i^*)} \cup F'$ contains at least $\varepsilon^{1/4}n^2/(2b^2)$ arcs having exactly one endpoint in $B_i^{(i^*)}$.*

Proof. Let $I_1, I_2, I_3 \subset \mathbb{R}$ be the intervals given by Lemma 4.6. Let $v \in [n] \setminus B_i^{(i^*)}$, such that $f'(v) \in I_2$. Pick $j \in [b]$, such that $v \in B_j^{(i^*)}$. We consider two cases:

Case 1: Suppose that $j < i$. Let $u \in B_i^{(i^*)}$, such that $f'(u) \in I_1$. If $(v,u) \in A^{(i^*)}$, then it follows that f' violates (x_{i^*}, v, u), and thus $(v,u) \in F'$. Otherwise, we have $(u,v) \in A^{(i^*)}$. Since u appears after v in the topological sort of $G^{(i^*)} \setminus F^{(i^*)}$, it follows that $(u,v) \in F^{(i^*)}$. Thus, in either case, $F^{(i^*)} \cup F'$ contains either (u,v) or (v,u). Therefore, $F^{(i^*)} \cup F'$ contains at least $n\varepsilon^{1/8}/(2b)$ arcs having u as an endpoint.

Case 2: Suppose that $j > i$. This case is similar to Case 1, and is included for completeness. Let $u \in B_i^{(i^*)}$, such that $f'(u) \in I_3$. If $(u,v) \in A^{(i^*)}$, then it follows that f' violates (x_{i^*}, u, v), and thus $(u,v) \in F'$. Otherwise, we have $(v,u) \in A^{(i^*)}$. Since u appears before v in the topological sort of $G^{(i^*)} \setminus F^{(i^*)}$, it follows that $(v,u) \in F^{(i^*)}$. Thus, in either case, $F^{(i^*)} \cup F'$ contains either (u,v) or (v,u). Therefore, $F^{(i^*)} \cup F'$ contains at least $n\varepsilon^{1/8}/(2b)$ arcs having u as an endpoint.

We conclude that, in either case, for any $u \in B_i^{(i^*)}$, $F^{(i^*)} \cup F'$ contains at least $n\varepsilon^{1/8}/(2b)$ arcs having u as an endpoint. Summing over all $u \in B_i^{(i^*)}$, we obtain that $F^{(i^*)} \cup F'$ contains at least $\varepsilon^{1/4}n^2/(2b^2)$ arcs having an endpoint in $B_i^{(i^*)}$. This concludes the proof. ◀

Next, we bound the number of unstable buckets.

▶ **Lemma 4.8.** *Let $J = \{i \in [b] : i \text{ is unstable}\}$, we have $|J| \leq O(\varepsilon^{1/4})2b^2$.*

Proof. By Lemma 4.3 we have that x_{i^*} is $(1 - O(\varepsilon^{1/2}))$-good in f', and by Lemma 4.4 we have that $G^{(i^*)}$ admits a feedback arc set of size at most $(O(\varepsilon^{1/2}))\binom{n-1}{2}$. Thus, by Theorem 3.3, the algorithm computes some feedback arc set $F^{(i^*)} \subset A^{(i^*)}$, with $|F^{(i^*)}| = O(\varepsilon^{1/2}n^2)$. We note that here we only use Theorem 3.3 to obtain a $O(1)$-approximation. By Lemma 4.7,

$$\begin{aligned} |J| &\leq |F^{(i^*)} \cup F'|/(\varepsilon^{1/4}n^2/(2b^2)) \\ &\leq O(\varepsilon^{1/4})2b^2, \end{aligned}$$

which concludes the proof. ◀

For any stable $i \in [b]$, let $\mathcal{I}_i \subset \mathbb{R}$ be the interval that contains at least $(1 - \varepsilon^{1/8})n/b$ points in $f'(B_i^{(i^*)})$, and at most $\varepsilon^{1/8}n/b$ other points. Let also $\mathcal{J}_i \subset \mathcal{I}_i$ be an open interval that contains all but the $\varepsilon^{1/8}n/b$ leftmost points in $f'(B_i^{(i^*)}) \cap \mathcal{I}_i$, and the $\varepsilon^{1/8}n/b$ rightmost points in $f'(B_i^{(i^*)}) \cap \mathcal{I}_i$. Thus, $|\mathcal{J}_i \cap f'(B_i^{(i^*)})| \geq (1 - 3\varepsilon^{1/8})n/b$. It follows that for any $i \neq j \in [b]$, such that both i and j are stable, we have $\mathcal{J}_i \cap \mathcal{J}_j = \emptyset$.

Intuitively, we intend to find a solution that satisfies a nearly-optimal fraction of constraints, while ignoring all constraints that involve points that are mapped outside the intervals $\mathcal{J}_i$, where $i \in [b]$ is stable. To that end, we define a small set of points that the analysis can safely "ignore":

$$X_{\mathsf{Noise}} = \bigcup_{i \in [b] : i \text{ stable}} \left\{ v \in B_i^{(i^*)} : f'(v) \notin \mathcal{J}_i \right\}.$$

Since $|\mathcal{J}_i \cap f'(B_i^{(i^*)})| \geq (1 - 3\varepsilon^{1/8})n/b$, it follows that

$$|X_{\mathsf{Noise}}| \leq 3\varepsilon^{1/8} n \tag{3}$$

Let also, for any $i \in [b]$,

$$\bar{B}_i^{(i^*)} = B_i^{(i^*)} \setminus X_{\mathsf{Noise}}.$$

We identify a set of triples $(i, j, k) \in [b]^3$ for which, intuitively, it is difficult to satisfy at least some significant fraction of all constraints with one point from each of the clusters $B_i^{(i^*)}$, $B_j^{(i^*)}$, and $B_k^{(i^*)}$. Formally, we say that some $(i, j, k) \in [b]^3$ is *brittle* if there exist $u, u' \in \mathcal{J}_i$, $v, v' \in \mathcal{J}_j$, and $w, w' \in \mathcal{J}_k$, such that

$$|u - v| < |u - w|,$$

and

$$|u' - v'| > |u' - w'|.$$

Intuitively, the above property implies that if for all $t \in [b]$, all points in $\bar{B}_t^{(i^*)}$ get mapped to the same point $p_t \in \mathcal{J}_t$, then there exist choices for the points $\{p_t\}_t$, such that some constraint in $\bar{B}_i^{(i^*)} \times \bar{B}_j^{(i^*)} \times \bar{B}_k^{(i^*)}$ is violated; in other words, if a triple (i, j, k) is not brittle, then the choice of the points p_t does not affect the satisfiability of the constraints in $\bar{B}_i^{(i^*)} \times \bar{B}_j^{(i^*)} \times \bar{B}_k^{(i^*)}$.

We are now ready to show that the retraction computed by the algorithm admits a solution of low total cost.

▶ **Lemma 4.9.** *The instance $\psi^{(i^*)}$ of WLLOC constructed in Step 4 admits a solution that satisfies constraints of total weight at least $|\mathcal{T}| \cdot (1 - O(\varepsilon^{1/8}))$.*

Proof. We define a mapping $g : [b] \to [0, 1]$, and $g' : [b] \to [0, 1]$, as follows. For each $i \in [b]$, pick $v_i \in \mathcal{B}_i^{(i^*)}$, arbitrarily, and set

$$g'(i) = f'(v_i).$$

For any $j \in [b]$, we set

$$g(j) = g'(i),$$

where $i \in [b]$ is the unique integer such that $i \in B_i^{(i^*)}$. By the definition of the WLLOC instance $\psi^{(i^*)}$, the total weight of the constraints violated by g' equals the total number of constraints violated by g. It therefore suffices to upper bound the number of constraints in $\mathcal{T}$ that are violated by g.

We define a partition $\mathcal{T} = \mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 \cup \mathcal{T}_4 \cup \mathcal{T}_5$, where

$$\mathcal{T}_0 = \{(u, v, w) \in \mathcal{T} : f' \text{ violates } (u, v, w)\},$$
$$\mathcal{T}_1 = \{(u, v, w) \in \mathcal{T} : \text{ at least two of } u, v, w \text{ are in the same cluster in } \mathcal{B}^{(i^*)}\},$$
$$\mathcal{T}_2 = \{(u, v, w) \in \mathcal{T} : u \in B_i^{(i^*)}, v \in B_j^{(i^*)}, w \in B_k^{(i^*)}, \text{ at least one of } i, j, k \text{ is unstable}\},$$
$$\mathcal{T}_3 = \{(u, v, w) \in \mathcal{T} : u \in B_i^{(i^*)}, v \in B_j^{(i^*)}, w \in B_k^{(i^*)}, \text{ and } (i, j, k) \text{ is brittle}\},$$
$$\mathcal{T}_4 = \{(u, v, w) \in \mathcal{T} : \{u, v, w\} \cap X_{\mathsf{Noise}} \neq \emptyset\},$$
$$\mathcal{T}_5 = \mathcal{T} \setminus (\mathcal{T}_0 \cup \mathcal{T}_1 \cup \mathcal{T}_2 \cup \mathcal{T}_3 \cup \mathcal{T}_4).$$

By Lemma 4.2 we have

$$|\mathcal{T}_1| \leq (\varepsilon + O(1/n))n\binom{n-1}{2}.$$

Since every cluster in $\mathcal{B}^{(i^*)}$ has n/b points, we have

$$|\mathcal{T}_1| \leq 3n^3/b^2. \tag{4}$$

In order to bound $|\mathcal{T}_3|$ we need a bound on the number of brittle triples. This is done in Lemma 4.8, which appears in Section 5. We thus have

$$|\mathcal{T}_2| \leq O(\varepsilon^{1/4})2n^3 b. \tag{5}$$

By Lemma 5.5 we have

$$|\mathcal{T}_3| \leq n^3/b. \tag{6}$$

By (3) we have

$$|\mathcal{T}_4| \leq O(\varepsilon^{1/8})n^3 \tag{7}$$

Let $(u, v, w) \in \mathcal{T}_5$. By the definition of $\mathcal{T}_5$, we have that $u \in \bar{B}_i^{(i^*)}$, $v \in \bar{B}_j^{(i^*)}$, and $w \in \bar{B}_k^{(i^*)}$, for some distinct $i, j, k \in [b]$, such that (i, j, k) is not brittle, and f' satisfies (u, v, w), that is

$$|f'(u) - f'(v)| < |f'(u) - f'(w)|.$$

By the definition of a brittle triple we get

$$|g(u) - g(v)| < |g(u) - g(w)|,$$

and thus g satisfies (u, v, w). We obtain that g satisfies all constraints in $\mathcal{T}_5$. Thus, by (4)–(7), the number of constraints violated by g is at most $|\mathcal{T}_0| + \ldots + |\mathcal{T}_4| \leq n^3 O(\varepsilon^{1/8})$, which concludes the proof. ◀

We are now ready to prove our main result.

Proof of Theorem 1.1. By Lemma 4.9 we have that WLLOC instance $\psi^{(i^*)} = ([b], \mathcal{T}', w)$ constructed at Step 4 of the algorithm, admits a mapping $g' : [b] \to \mathbb{R}$, such that the total weight of the constraints in $\mathcal{T}'$ violated by g' is at most $O(\varepsilon^{1/8}n^3)$. Therefore, in Step 5, using the exact algorithm from Theorem 3.1, we compute a mapping $g : [b] \to \mathbb{R}$, violating the same total weight as g'. By the definition of retraction, it follows that the mapping $f^{(i^*)}$ computed in Step 5 violates at most $O(\varepsilon^{1/8}n^3)$ constraints in $\mathcal{T}$, as required.

It remains to bound the running time. Step 2 uses the algorithm from Theorem 3.3 to obtain a $O(1)$-approximate minimum feedback arc set, and thus takes time $O(n^6)$. Step 3 takes time $O(n)$ and Step 4 takes time $O(n^2)$. Step 5 runs the algorithm from Theorem 3.1 on an input of size b, and thus takes time $O(b^{3b}) + O(n)$. Step 6 requires computing the number of violated constraints in each of the n solutions, and thus takes total time $O(n^4)$. Due to Step 1, the Steps 2–5 are repeated n times, and thus the total running time is at most $O(n^7 + b^{3b}n) = O(n^7) + (1/\varepsilon)^{O(1/\varepsilon^{1/8})}n$, which concludes the proof. ◀

5 Bounding the number of brittle triples

This Section is devoted to proving an upper bound on the number of brittle triples. We begin by deriving a simple condition that is a consequence of brittleness.

▶ **Lemma 5.1.** *Let $j < i < k \in [b]$. We have that if (i, j, k) is brittle, then there exist $p_i \in \mathcal{J}_i$, $p_j \in \mathcal{J}_j$, $p_k \in \mathcal{J}_k$, such that*

$$p_i - p_j = p_k - p_i.$$

Proof. If (i, j, k) is brittle, it is easy to see that $\mathcal{J}_i$ must be located between $\mathcal{J}_j$ and $\mathcal{J}_k$; otherwise, any representative point chosen in $\mathcal{J}_i$ must be closer to all the points in $\mathcal{J}_j$ than those in $\mathcal{J}_k$, or vice versa. By definition, there exist $p_i \in \mathcal{J}_i$, $p'_j \in \mathcal{J}_j$, $p'_k \in \mathcal{J}_k$, such that

$$p_i - p'_j \geq p'_k - p_i,$$

and $p''_j \in \mathcal{J}_j$, $p''_k \in \mathcal{J}_k$, such that

$$p_i - p''_j < p''_k - p_i.$$

Without loss of generality, assume $p'_j < p''_j$ and $p'_k < p''_k$, and define $\delta \in [0, 1]$. Comparing $d_{ij}(\delta) = p_i - (p'_j + \delta(p''_j - p'_j))$ and $d_{ik}(\delta) = (p'_k + \delta(p''_k - p'_k)) - p_i$, we have $d_{ij}(0) - d_{ik}(0) \geq 0$ and $d_{ij}(1) - d_{ik}(1) < 0$. There exist $\delta' \in [0, 1]$, s.t. $d_{ij}(\delta') - d_{ik}(\delta') = 0$.

Define $p_j = (p'_j + \delta'(p''_j - p'_j)) \in \mathcal{J}_i$ and $p_k = (p'_k + \delta'(p''_k - p'_k)) \in \mathcal{J}_j$, we have

$$p_i - p_j = p_k - p_i,$$

which concludes the proof. ◀

▶ **Lemma 5.2.** *Let $i_1, i_2, i_3, j_1, j_2, j_3, k_1, k_2, k_3 \in \mathbb{R}$, with $i_1 < i_2 < i_3$, $j_1 < j_2 < j_3$, $k_1 < k_2 < k_3$. For any $\alpha, \beta, \gamma \in \{1, 2\}$, let $H_{\alpha,\beta,\gamma}$ be the axis-parallel parallelepiped defined by*

$$H_{\alpha,\beta,\gamma} := \mathcal{CH}(\{(i_{\alpha+\alpha'}, j_{\beta+\beta'}, k_{\gamma+\gamma'}) : \alpha', \beta', \gamma' \in \{0, 1\}\}).$$

Let h be any plane in $\mathbb{R}^3$. Then, there exist $\alpha^*, \beta^*, \gamma^* \in \{0, 1\}$, such that h does not intersect the interior of $H_{\alpha^*,\beta^*,\gamma^*}$.

Proof. For any $d \geq 2$, any d-dimensional halfspace containing the origin must also contain at least one d-orthant. The assertion follows immediately from the case $d = 3$. ◀

▶ **Lemma 5.3.** *Let $i, j, k \in [b]$, with $j+1 < i$, and $i+1 < k$. Then, there exist $i', j', k' \in \{0, 1\}$ such that $(i + i', j + j', k + k')$ is not brittle.*

Proof. Define the plane

$$h = \{(x_I, x_J, x_K) \in \mathbb{R}^3 : x_I - x_J = x_K - x_I\}.$$

By Lemma 5.1, we have that if $(i + i', j + j', k + k')$ is brittle, then h must intersect the hyperrectangle $\mathcal{J}_{i+i'} \times \mathcal{J}_{j+j'} \times \mathcal{J}_{k+k'}$. However, by Lemma 5.2, it follows that there exist $i', j', k' \in \{0, 1\}$, such that h does not intersect $\mathcal{J}_{i+i'} \times \mathcal{J}_{j+j'} \times \mathcal{J}_{k+k'}$, and thus $(i + i', j + j', k + k')$ is not brittle, which concludes the proof. ◀

▶ **Lemma 5.4** (Brittle convexity)**.** *Let $\{e_1, e_2, e_3\}$ be the standard orthonormal basis in $\mathbb{R}^3$. Let $v \in [b-2]^3$, and let $w \in \{e_1, e_2, e_3\}$, such that v and $v + 2w$ are both brittle. Then, $v + w$ is also brittle.*

Proof. By Lemma 5.1, there exist $p_i \in \mathcal{J}_i$, $p_j \in \mathcal{J}_j$, $p_k \in \mathcal{J}_k$, such that

$$p_i - p_j = p_k - p_i. \tag{8}$$

Let $w = (i', j', k')$. Similarly, there exist $q_i \in \mathcal{J}_{i+2i'}$, $q_j \in \mathcal{J}_{j+2j'}$, $q_k \in \mathcal{J}_{k+2k'}$, such that

$$q_i - q_j = q_k - q_i. \tag{9}$$

For any $\alpha \in [0,1]$, let

$$z_i^{(\alpha)} = (1-\alpha)p_i + \alpha q_i,$$

$$z_j^{(\alpha)} = (1-\alpha)p_j + \alpha q_j,$$

$$z_k^{(\alpha)} = (1-\alpha)p_k + \alpha q_k.$$

Let us assume that $w = e_1$. The cases $w = e_2$ and $w = e_3$ can be handled in a similar manner. We have that for all $\alpha \in [0,1]$, $z_j^{(\alpha)} \in \mathcal{J}_j$, and $z_k^{(\alpha)} \in \mathcal{J}_k$. Moreover, $z_i^{(0)} \in \mathcal{J}_i$, and $z_i^{(1)} \in \mathcal{J}_{i+2}$, which implies that there exists some $\alpha^* \in [0,1]$, such that $z_i^{(\alpha^*)} \in \mathcal{J}_{i+1}$. We have

$$\begin{aligned} z_i^{(\alpha^*)} - z_j^{(\alpha^*)} &= (1-\alpha^*)p_i + \alpha^* q_i - (1-\alpha^*)p_j - \alpha^* q_j \\ &= (1-\alpha^*)(p_i - p_j) + \alpha^*(q_i - q_j) \\ &= (1-\alpha^*)(p_k - p_i) + \alpha^*(q_k - q_i) \\ &= (1-\alpha^*)p_k + \alpha^* q_k - (1-\alpha^*)p_i - \alpha^* q_i \\ &= z_k^{(\alpha^*)} - z_i^{(\alpha^*)}, \end{aligned}$$

which by Lemma 5.1 implies that $v + w$ is brittle, and concludes the proof. ◀

We are now ready to bound the number of brittle triples, which is the main result of this Section.

▶ **Lemma 5.5.** *The number of brittle triples is at most $O(b^2)$.*

Proof. Let $B \subseteq [b]^3$ be the set of all brittle triples, and let $B' = [b]^3 \setminus B$. For any $s \in \{0,1\}^3$, let

$$U_s = s \cdot b/2 + [b/2]^3,$$

and $B_s = B \cap U_s$. Since $B = \bigcup_s B_s$, and there are only 8 different values for s, it suffices to show that for any $s \in \{0,1\}^3$, $|B_s| = O(b^2)$. We shall prove this for the case $s = (0,0,0)$. All remaining cases can be handled in a similar manner.

For the remainder for the proof, let $s = (0,0,0)$. By Lemma 5.3, it follows that for any $v \in B_s^3$, there exists $v' \in B'$, with $v' - v \in \{0,1\}^3$. This implies that there exists $u \in B$, and $u' \in B'$, with $u - v \in \{0,1\}^3$, $u' - v \in \{0,1\}^3$, and $u' - u \in \{e_1, e_2, e_3\}$, where $\{e_1, e_2, e_3\}$ is the standard orthonormal basis in $\mathbb{R}^3$. Let $t = u' - u$. By Lemma 5.4, it follows by induction that for any $i \in \{1, \ldots, b/2\}$, the triple $u + i \cdot t$ is brittle. Let

$$R_v = \bigcup_{i=1}^{b/2} \{u + c \cdot i\}.$$

Thus $R_v \subseteq B'$. Note that, since $s = (0,0,0)$, we have

$$|R_v| \geq b/2. \tag{10}$$

For any $j \in \{1,2,3\}$, we say that v is *type-j*, if $t = e_j$.

Let

$$B_{s,j} = \{v \in B_s : v \text{ is type-}j\}.$$

Let $j^* \in \{1,2,3\}$, such that $|B_{s,j^*}| \geq |B_s|/3$.

By the above construction, it follows that for any $v, w \in B_{s,j^*}$, with $\|v - w\|_\infty \geq 2$, we have $R_v \cap R_w = \emptyset$. We greedily construct some $C \subseteq B_{s,j^*}$ as follows. We start with $C := \emptyset$, and $D := B_{s,j^*}$. While $D \neq \emptyset$, we pick any $v \in D$, and we set $C := C \cup \{v\}$, and $D := D \setminus \mathsf{Ball}_\infty(v,1)$, where $\mathsf{Ball}_\infty(v,r)$ denotes the ℓ_∞-ball of radius r centered at v. For every v added to C, we delete at most 9 elements from D, and thus

$$|C| \geq |B_{s,j^*}|/9 \geq |B_s|/27.$$

Since for any $v, w \in C$, we have $\|v - w\|_\infty$, it follows that $R_v \cap R_w = \emptyset$. Combining with (10), we get

$$b^3 \geq |B'| \geq \left| \bigcup_{v \in C} R_v \right| = \sum_{v \in C} |R_v| \geq |C| \cdot b/2 \geq |B_s| \cdot b/54,$$

and thus $|B_s| \leq 54b^2$, which concludes the proof. ◀

References

1. Nir Ailon and Noga Alon. Hardness of fully dense problems. *Information and Computation*, 205(8):1117–1129, 2007.
2. Noga Alon, Mihai Bădoiu, Erik D Demaine, Martin Farach-Colton, MohammadTaghi Hajiaghayi, and Anastasios Sidiropoulos. Ordinal embeddings of minimum relaxation: general properties, trees, and ultrametrics. *ACM Transactions on Algorithms (TALG)*, 4(4):1–21, 2008.
3. Mihai Badoiu. Approximation algorithm for embedding metrics into a two-dimensional space. In *Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 434–443. Society for Industrial and Applied Mathematics, 2003.
4. Mihai Bădoiu, Erik D Demaine, MohammadTaghi Hajiaghayi, Anastasios Sidiropoulos, and Morteza Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 21–34. Springer, 2008.
5. Mihai Badoiu, Kedar Dhamdhere, Anupam Gupta, Yuri Rabinovich, Harald Räcke, Ramamoorthi Ravi, and Anastasios Sidiropoulos. Approximation algorithms for low-distortion embeddings into low-dimensional spaces. In *SODA*, volume 5, pages 119–128. Citeseer, 2005.
6. Mihai Bădoiu, Julia Chuzhoy, Piotr Indyk, and Anastasios Sidiropoulos. Low-distortion embeddings of general metrics into the line. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 225–233. ACM, 2005.
7. Timothy Carpenter, Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Anastasios Sidiropoulos. Algorithms for low-distortion embeddings into arbitrary 1-dimensional spaces. In *34th International Symposium on Computational Geometry (SoCG 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.
8. Moses Charikar, Venkatesan Guruswami, and Rajsekar Manokaran. Every permutation csp of arity 3 is approximation resistant. In *2009 24th Annual IEEE Conference on Computational Complexity*, pages 62–73. IEEE, 2009.
9. Benny Chor and Madhu Sudan. A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523, 1998.

10 Kedar Dhamdhere, Anupam Gupta, and R. Ravi. Approximation algorithms for minimizing average distortion. *Theory Comput. Syst.*, 39(1):93–111, 2006. doi:10.1007/s00224-005-1259-6.

11 Michael R Fellows, Fedor V Fomin, Daniel Lokshtanov, Elena Losievskaja, Frances A Rosamond, and Saket Saurabh. Distortion is fixed parameter tractable. In *International Colloquium on Automata, Languages, and Programming*, pages 463–474. Springer, 2009.

12 Diego Ihara, Neshat Mohammadi, Francesco Sgherzi, and Anastasios Sidiropoulos. Robust mahalanobis metric learning via geometric approximation algorithms. *CoRR*, abs/1905.09989, 2019. arXiv:1905.09989.

13 Diego Ihara, Neshat Mohammadi, and Anastasios Sidiropoulos. Algorithms for metric learning via contrastive embeddings. In *35th International Symposium on Computational Geometry (SoCG 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

14 Piotr Indyk, Jiří Matoušek, and Anastasios Sidiropoulos. Low-distortion embeddings of finite metric spaces. In Jacob E. Goodman, Joseph O'Rourke, and Csaba D. Toth, editors, *Handbook of Discrete and Computational Geometry, Second Edition.* Chapman and Hall/CRC, 2017.

15 Marek Karpinski and Warren Schudy. Approximation schemes for the betweenness problem in tournaments and related ranking problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 277–288. Springer, 2011.

16 Claire Kenyon-Mathieu and Warren Schudy. How to rank with few errors. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 95–103, 2007.

17 Brian Kulis et al. Metric learning: A survey. *Foundations and Trends® in Machine Learning*, 5(4):287–364, 2013.

18 Yury Makarychev. Simple linear time approximation algorithm for betweenness. *Operations research letters*, 40(6):450–452, 2012.

19 Amir Nayyeri and Benjamin Raichel. Reality distortion: Exact and approximate algorithms for embedding into the line. In *Foundations of Computer Science (FOCS), 2015 IEEE 56th Annual Symposium on*, pages 729–747. IEEE, 2015.

20 Amir Nayyeri and Benjamin Raichel. A treehouse with custom windows: Minimum distortion embeddings into bounded treewidth graphs. 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 724–736. SIAM, 2017. doi:10.1137/1.9781611974782.46.

21 Jaroslav Opatrny. Total ordering problem. *SIAM Journal on Computing*, 8(1):111–114, 1979.

22 Yuri Rabinovich. On average distortion of embedding metrics into the line and into l1. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 456–462, 2003.

23 Gregory Shakhnarovich. *Learning task-specific similarity.* PhD thesis, Massachusetts Institute of Technology, 2005.

24 Csaba D Toth, Joseph O'Rourke, and Jacob E Goodman. *Handbook of discrete and computational geometry.* Chapman and Hall/CRC, 2017.

Improved Circular k-Mismatch Sketches

Shay Golan
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
golansh1@cs.biu.ac.il

Tomasz Kociumaka
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
kociumaka@mimuw.edu.pl

Tsvi Kopelowitz
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
kopelot@gmail.com

Ely Porat
Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel
porately@cs.biu.ac.il

Przemysław Uznański
Institute of Computer Science, University of Wrocław, Poland
puznanski@cs.uni.wroc.pl

Abstract

The shift distance $\mathsf{sh}(S_1, S_2)$ between two strings S_1 and S_2 of the same length is defined as the minimum Hamming distance between S_1 and any rotation (cyclic shift) of S_2. We study the problem of sketching the shift distance, which is the following communication complexity problem: Strings S_1 and S_2 of length n are given to two identical players (encoders), who independently compute sketches (summaries) $\mathtt{sk}(S_1)$ and $\mathtt{sk}(S_2)$, respectively, so that upon receiving the two sketches, a third player (decoder) is able to compute (or approximate) $\mathsf{sh}(S_1, S_2)$ with high probability.

This paper primarily focuses on the more general k-mismatch version of the problem, where the decoder is allowed to declare a failure if $\mathsf{sh}(S_1, S_2) > k$, where k is a parameter known to all parties. Andoni et al. (STOC'13) introduced exact circular k-mismatch sketches of size $\tilde{O}(k + D(n))$, where $D(n)$ is the number of divisors of n. Andoni et al. also showed that their sketch size is optimal in the class of linear homomorphic sketches.

We circumvent this lower bound by designing a (non-linear) exact circular k-mismatch sketch of size $\tilde{O}(k)$; this size matches communication-complexity lower bounds. We also design $(1 \pm \varepsilon)$-approximate circular k-mismatch sketch of size $\tilde{O}(\min(\varepsilon^{-2}\sqrt{k}, \varepsilon^{-1.5}\sqrt{n}))$, which improves upon an $\tilde{O}(\varepsilon^{-2}\sqrt{n})$-size sketch of Crouch and McGregor (APPROX'11).

2012 ACM Subject Classification Theory of computation → Pattern matching; Theory of computation → Sketching and sampling

Keywords and phrases Hamming distance, k-mismatch, sketches, rotation, cyclic shift, communication complexity

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.46

Category APPROX

Funding This work was supported in part by ISF grants no. 1278/16 and 1926/19, by a BSF grant no. 2018364, and by an ERC grant MPM under the EU's Horizon 2020 Research and Innovation Programme (grant no. 683064).
Przemysław Uznański: Supported by Polish National Science Centre grant 2019/33/B/ST6/00298.


© Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, Ely Porat, and Przemysław Uznański;
licensed under Creative Commons License CC-BY



1 Introduction

The *Hamming distance* [25] is a fundamental metric for strings, and computing the Hamming distances in various settings is a central task in text processing. The Hamming distance of two length-n strings S_1 and S_2 is defined as the number of aligned mismatches between S_1 and S_2. In the k-mismatch variant [1, 4, 14, 22, 32], the problem is parameterized by an integer $1 \leq k \leq n$, and the task is relaxed so that if $\mathsf{Ham}(S_1, S_2) > k$, then instead of computing $\mathsf{Ham}(S_1, S_2)$, the algorithm is only required to report that this is the case, without computing the distance. Since computing the exact Hamming distance, both in the classic version and the k-mismatch version, is challenging under some efficiency constraints, a large body of research [14, 27, 30, 31] focused on the approximation version of both problems. Formally, in the $(1 \pm \varepsilon)$-approximation variant of either problem, the problem is parameterized by $\varepsilon > 0$, and whenever the algorithm should report $\mathsf{Ham}(S_1, S_2)$ in the original problem, in the approximation variant, the algorithm may report a $(1 \pm \varepsilon)$-approximation of $\mathsf{Ham}(S_1, S_2)$.

Sketching. Sketching is one of the settings of sublinear algorithms designed for space-efficient and time-efficient processing of big data, with applications in streaming algorithms, signal processing, network traffic monitoring, and other areas [18, 17, 34]. The task of sketching the Hamming distance boils down to constructing two (randomized) functions $\mathtt{sk} : \Sigma^n \to \{0,1\}^*$ and $\mathtt{dec} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$ such that $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) = \mathsf{Ham}(S_1, S_2)$ holds with high probability[1]. The communication-complexity interpretation of this problem involves three players sharing public randomness: two identical encoders and a decoder. The first encoder receives a string S_1, while the second encoder receives a string S_2. Each of the encoders needs to independently summarize its string. The summaries (sketches) are then sent to the decoder, whose task is to retrieve $\mathsf{Ham}(S_1, S_2)$ based on the summaries alone, without access to S_1 or S_2. The sketching complexity of Hamming distance, which is the size of the sketch, is well understood: the optimal sketch size is $\tilde{\Theta}(n)$ for the base variant [37, 40], $\tilde{\Theta}(k)$ for the k-mismatch variant [26, 37], and $\tilde{\Theta}(\varepsilon^{-2})$ for the $(1 \pm \varepsilon)$-approximate variants [2, 33, 40].[2] Much less is known about the sketching complexity of edit distance: it is $\tilde{\Theta}(n)$ for the base variant and $\tilde{O}(k^8)$ for the k-error variant [9]. Approximate edit distance sketches with super-constant approximation ratios are also known; see e.g. [11, 35].

The shift distance. We consider the *shift distance* [5, 6, 19], which is a cyclic variant of Hamming distance. For two strings $S_1, S_2 \in \Sigma^n$, the shift distance is defined as the minimum Hamming distance between S_1 and any cyclic shift (rotation) of S_2. Formally, if cyc is a function cyclically shifting a given string (by one position to the left), then $\mathsf{sh}(S_1, S_2) = \min\{\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \mid m \in \mathbb{Z}\}$ is the shift distance between S_1 and S_2. The research on shift distance for sublinear algorithms is mostly motivated by the observation that the shift distance shares many similarities with the fundamental Hamming distance. At the same time, shift distance inherits some of the challenges exhibited in the edit distance, e.g., in the context of low-dimensional embeddings to ℓ_1 [29] and asymmetric query complexity [7].

The first sketching scheme for shift distance, by Andoni et al. [6], allows for $O(\log^2 n)$-approximation using sketches of size $\tilde{O}(1)$. Crouch and McGregor [19] showed $(1 \pm \varepsilon)$-approximate sketches for shift distance that use $\tilde{O}(\varepsilon^{-2}\sqrt{n})$ space. Andoni et al. [5] designed

[1] An event $\mathcal{E}$ is said to happen with high probability if $\Pr[\mathcal{E}] \geq 1 - n^{-\Omega(1)}$.

[2] Throughout this paper, the $\tilde{\Theta}(\cdot), \tilde{\Omega}(\cdot)$, and $\tilde{O}(\cdot)$ notations suppress $\log^{O(1)} n$ factors.

exact k-mismatch circular sketches that use $\tilde{O}(D(n)+k)$ space, where $D(n)$ is the number of divisors of n, which is $n^{\Theta(1/\log\log n)}$ in the worst case. In [5], it is proven that $\tilde{\Omega}(D(n))$ is a lower bound for any linear homomorphic sketch for the shift distance k-mismatch problem.[3]

Our results. We consider a (slight) generalization of the problem of sketching the shift distance, where the decoder needs to retrieve $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ for every $m \in \mathbb{Z}$. We consider the problem both in the exact setting and in the $(1 \pm \varepsilon)$-approximation version.

▶ **Problem 1.1.** An *exact circular k-mismatch sketch* (k-ECS) for $\Pi \subseteq \Sigma^n$ is a pair of randomized functions[4] $\mathtt{sk} : \Pi \to \{0,1\}^*$ and $\mathtt{dec} : \{0,1\}^* \times \{0,1\}^* \times \mathbb{Z} \to \mathbb{N}$ such that, for every $S_1, S_2 \in \Pi$ and $m \in \mathbb{Z}$, the following holds with high probability:
- if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \le k$, then $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$,
- otherwise, $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) > k$.

▶ **Problem 1.2.** A *$(1 \pm \varepsilon)$-approximate circular k-mismatch sketch* ((ε, k)-ACS) for $\Pi \subseteq \Sigma^n$ is a pair of randomized functions $\mathtt{sk} : \Pi \to \{0,1\}^*$ and $\mathtt{dec} : \{0,1\}^* \times \{0,1\}^* \times \mathbb{Z} \to \mathbb{R}$ such that, for every $S_1, S_2 \in \Pi$ and $m \in \mathbb{Z}$, the following holds with high probability:
- if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \le k$, then $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) \in (1 \pm \varepsilon)\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$,
- otherwise, $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) > (1-\varepsilon)k$.

In this paper, a sketch for $\Pi \subseteq \Sigma^n$ is of size s if for every $S \in \Pi$, we have $|\mathtt{sk}(S)| \le s$ with high probability. Our results are stated in the following theorems.

▶ **Theorem 1.3.** *There exists a k-ECS sketch for Σ^n of size $\tilde{O}(k)$.*

▶ **Theorem 1.4.** *There exists an (ε, k)-ACS sketch for Σ^n of size $\tilde{O}(\min(\varepsilon^{-2}\sqrt{k}, \varepsilon^{-1.5}\sqrt{n}))$.*

Notice that Theorem 1.3 circumvents the lower bound of Andoni et al. [5] by using non-linear sketches (however, the sketches are still homomorphic). Moreover, Theorem 1.4 improves upon the $\tilde{O}(\varepsilon^{-2}\sqrt{n})$ size sketches of Crouch and McGregor [19], and also addresses the more general k-mismatch variant of the problem.

Decoding efficiency. We also discuss the efficiency of evaluating $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m)$ for a given $m \in \mathbb{Z}$ and the efficiency of evaluating or approximating $\mathsf{sh}(S_1, S_2)$ based on our sketches. We show that the naive solution of minimizing $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m)$ across all $m \in [n]$ can be sped up significantly. Formally, this yields solutions to the following problems.

▶ **Problem 1.5.** An *exact k-mismatch shift distance sketch* (k-ESDS) for $\Pi \subseteq \Sigma^n$ is a pair of randomized functions $\mathtt{sk} : \Pi \to \{0,1\}^*$ and $\mathtt{dec}^{\mathsf{sh}} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{N}$ such that, for every $S_1, S_2 \in \Pi$, the following holds with high probability:
- if $\mathsf{sh}(S_1, S_2) \le k$, then $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) = \mathsf{sh}(S_1, S_2)$,
- otherwise, $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) > k$.

▶ **Problem 1.6.** A *$(1 \pm \varepsilon)$-approximate k-mismatch shift distance sketch* ((ε, k)-ASDS) for $\Pi \subseteq \Sigma^n$ is a pair of randomized functions $\mathtt{sk} : \Pi \to \{0,1\}^*$ and $\mathtt{dec}^{\mathsf{sh}} : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ such that, for every $S_1, S_2 \in \Pi$, the following holds with high probability:
- if $\mathsf{sh}(S_1, S_2) \le k$, then $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) \in (1 \pm \varepsilon)\mathsf{sh}(S_1, S_2)$,
- otherwise, $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) > (1-\varepsilon)k$.

The task of designing efficient algorithms for computing our sketches is left open.

[3] A sketch is *homomorphic* if $\mathtt{sk}(\mathsf{cyc}(S))$ can be retrieved from $\mathtt{sk}(S)$ and *linear* if $\mathtt{sk}$ is a linear mapping.
[4] A randomized function $f : X \to Y$ is a random variable whose values are functions from X to Y.

Related work. A problem closely related to the *circular Hamming distances* problem, asking to determine $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ for all $0 \leq m < n$, is the *text-to-pattern Hamming distances* problem, where the input consists of a pattern P (of length m) and a text T (of length n), and the task is to compute the Hamming distances between P and every length-m substring of T. A straightforward reduction from the circular Hamming distances problem to the text-to-pattern Hamming distances problem is given by $P = S_1$ and $T = S_2 \cdot S_2$.

In the offline setting, including the exact and approximate k-mismatch variants, we are not aware of any separation between the two problems. The state-of-the-art exact solution combines an $\tilde{O}(n\sigma)$-time solution for small alphabets (of size σ) [21] with an $\tilde{O}(n + \frac{nk}{\sqrt{m}})$-time algorithm [22], which culminates a long line of research [1, 4, 14, 32]. The approximate variant can be solved in $\tilde{O}(\varepsilon^{-1}n)$ time [30, 31]; these results improve upon [27]. On the other hand, sketches for text-to-pattern Hamming distances need to be much larger than circular sketches: already recovering exact occurrences requires $\Omega(n - m)$ space [8].

Interestingly, both in the exact and in the approximate setting, the sizes of our circular k-mismatch sketches coincide with the current upper bounds for space usage in the *streaming* k-mismatch problem. In that model, the text arrives in a stream, one character at a time, and the goal is to compute, or estimate, after the arrival of each text character, the Hamming distance between P and the current suffix of T. The state-of-the-art exact algorithm [15] uses $\tilde{O}(k)$ space and costs $\tilde{O}(\sqrt{k})$ time per character, which improves upon [14, 23, 36, 38]. A recent approximate streaming algorithm [12] uses $\tilde{O}(\min(\varepsilon^{-2}\sqrt{k}, \varepsilon^{-1.5}\sqrt{n}))$ space and costs $\tilde{O}(\varepsilon^{-3})$ time per character, which improves upon [16, 39].

2 Algorithmic Overview and Organization

The central technical contribution of our work is a randomized scheme of selecting positions in a given string $S \in \Sigma^n$ so that if $f(S) \subseteq \{1, \ldots, n\}$ is the set of selected positions, then the following properties hold: $|f(S)| = \tilde{O}(k)$ with high probability, the selection is preserved by rotations (the selected positions are shifted along with the underlying characters), and $|f(S) \cap f(T)| \geq k$ with high probability for every $T \in \Sigma^n$ such that $\mathsf{Ham}(S, T) \leq k$.

Unfortunately, for integer exponents $\alpha \gg k$, such a selection of positions is infeasible for strings of the form $S = Q^\alpha$ (that we call *high powers*), which are fixed points of $\mathsf{cyc}^{n/\alpha}$. Moreover, the selection of positions is also infeasible for strings with a relatively small Hamming distance to some high power. Hence, we define the problematic strings to be *pseudo-periodic*, exclude them from the selection scheme, and deal with them separately.

Sketches for non-pseudo-periodic strings. In Section 4, we construct sketches for non-pseudo-periodic strings using a selection function f satisfying the aforementioned properties.

Our (ε, k)-ACS sketch stores (non-circular) approximate Hamming distance sketches of $\mathsf{cyc}^i(S)$ for a random sample of $\tilde{O}(\sqrt{k})$ positions $i \in f(S)$. Given the (ε, k)-ACS sketches of two strings S_1, S_2 and a shift value m such that $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, with high probability, there is a shift i such that the non-circular sketches of both $\mathsf{cyc}^i(S_1)$ and $\mathsf{cyc}^{i+m}(S_2)$ are available. The decoder uses these approximate Hamming distance sketches to approximate $\mathsf{Ham}(\mathsf{cyc}^i(S_1), \mathsf{cyc}^{i+m}(S_2)) = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$; see Section 4.1 for details.

Our k-ECS sketch, for each position $i \in f(S)$, stores a (non-circular) sketch of $\mathsf{cyc}^i(S)$ capable of retrieving each mismatch with probability $\Theta(\frac{\log n}{k})$, but no more than $O(\log n)$ mismatches in total. Given circular sketches of two strings S_1, S_2 and a shift value m such that $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, with high probability, there are at least k shifts i such that the non-circular sketches of both $\mathsf{cyc}^i(S_1)$ and $\mathsf{cyc}^{i+m}(S_2)$ are available. Each of these k pairs of

non-circular sketches yields random mismatches between S_1 and $\mathsf{cyc}^m(S_2)$. Consequently, with high probability, each mismatch between S_1 and $\mathsf{cyc}^m(S_2)$ is reported at least once, which allows for the exact computation of $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$; see Section 4.2 for details.

Selection function. The selection function f for non-pseudo-periodic strings is constructed in Section 5. Our baseline solution is to sample strings of length $\frac{n}{\gamma k}$ (for a constant γ fixed in Section 4) with rate $\tilde{O}(\frac{k}{n})$ and, for each sampled string u, to add to $f(S)$ the positions where u occurs in S. Unfortunately, since substrings could have much more than γk occurrences, the variance of $|f(S)|$ could be rather large, and thus substrings with a large number of occurrences need to be excluded from the sample. This workaround is feasible unless highly periodic regions cover most positions of S; see Section 5.1, where the properties of f are proved using concentration arguments (the Chernoff–Hoeffding bound).

In the complementary case of strings mostly covered by highly periodic regions, we utilize the structure of these regions to deterministically select positions. If there are many disjoint regions, it suffices to select the boundaries of the regions. However, in general we follow a more involved approach inspired by [10, 13]: periodic regions are extended as long as the number of mismatches between the extended region and the period of the region is relatively small compared to the length of the extended region. The positions of these mismatches are also added to $f(S)$. Selection of f in this case is the most technically challenging component of our construction; see Section 5.2 for details.

Sketches for pseudo-periodic strings. Each pseudo-periodic string can be assigned to the nearest high power (the *base*) so that two pseudo-periodic strings S_1, S_2 satisfy $\mathsf{Ham}(S_1, S_2) \leq k$ only if they share the same base. Thus, we first design a 0-mismatch circular sketch (of size $\tilde{O}(1)$) to be used for comparing the bases both in the exact and approximate variants.

Our exact k-mismatch circular sketch stores the mismatches between the string and its base. Once the decoder verifies that S_1 and $\mathsf{cyc}^m(S_2)$ share the same base, the mismatches between S_1 and $\mathsf{cyc}^m(S_2)$ are reconstructed from the mismatches between each of the strings $S_1, \mathsf{cyc}^m(S_2)$ and their common base. The (ε, k)-ACS sketch stores only the mismatches between the string and its base at $\tilde{O}(\frac{n}{\varepsilon\sqrt{k}})$ sampled positions (so that $\tilde{O}(\varepsilon^{-1}\sqrt{k})$ mismatches are stored with high probability). Once the decoder verifies that S_1 and $\mathsf{cyc}^m(S_2)$ share the same base, the mismatches between S_1 and $\mathsf{cyc}^m(S_2)$ at $\tilde{O}(\frac{n}{\varepsilon^2 k})$ jointly sampled positions are retrieved to estimate $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$; see Section 6.

Organization. In Sections 4 and 5, we describe the main novel ideas and techniques of this paper, which are used in sketches for strings that are not pseudo-periodic. In Section 6, we provide sketches for pseudo-periodic strings, and in Section 7 we combine the sketches of Section 4 with the sketches of Section 6 in order to prove the main theorems. Notice that these two cases require a slight overlap so that whenever $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, one of the cases accommodates *both* S_1 and S_2. In Section 7, we also develop another (ε, k)-ACS sketch, tailored to approximating large distances. This simple construction improves the size of (ε, k)-ACS sketches (for $k \geq \varepsilon n$) from $\tilde{O}(\varepsilon^{-2}\sqrt{k})$ to $\tilde{O}(\varepsilon^{-1.5}\sqrt{n})$. Finally, in Appendix A, we describe efficient decoding algorithms for retrieving the shift distance from the encodings developed for the circular k-mismatch sketches. A few simple or folklore proofs are deferred from Sections 3–7 to Appendix B.

3 Preliminaries

For integers $\ell \leq r$, we denote $[\ell\,..\,r] = \{\ell, \ell+1, \ldots, r\}$. Moreover, $[n] = [1\,..\,n]$.

A string S of length $|S| = n$ is a sequence of characters $S[1]S[2]\cdots S[n]$ over an alphabet Σ; in this work, we assume that $\Sigma = [\sigma]$. The set of all length-n strings over Σ is denoted by Σ^n. A string T is a *substring* of a string $S \in \Sigma^n$ if $T = S[i]S[i+1]\cdots S[j]$ for $1 \leq i \leq j \leq n$. In this case, we denote the occurrence of T at position i by $S[i\,..\,j]$. Such an occurrence is a *fragment* of S. A fragment $S[i\,..\,j]$ is a *prefix* of S if $i = 1$ and a *suffix* of S if $j = n$.

Hamming distance. The *Hamming distance* $\mathsf{Ham}(S,T)$ of two strings $S, T \in \Sigma^n$ is defined as the number of positions $i \in [n]$ such that $S[i] \neq T[i]$. We denote $\mathsf{MP}(S,T) = \{i \in [n] \mid S[i] \neq T[i]\}$ to be the set of *mismatch positions* and $\mathsf{MI}(S,T) = \{(i, S[i], T[i]) \mid i \in [n], S[i] \neq T[i]\}$ to be the underlying *mismatch information*. Note that $\mathsf{Ham}(S,T) = |\mathsf{MP}(S,T)| = |\mathsf{MI}(S,T)|$.

For a subset $A \subseteq [n]$, we denote $\mathsf{MI}_A(S,T) = \{(i,a,b) \in \mathsf{MI}(S,T) \mid i \in A\}$ and $\mathsf{Ham}_A(S,T) = |\mathsf{MI}_A(S,T)|$. The following result, based on the Chernoff bound and proved in Appendix B, shows that $\mathsf{Ham}_A(S,T)$ for random A yields an approximation of $\mathsf{Ham}(S,T)$.

▶ **Lemma 3.1.** *Let A be a random subset of $[n]$ with elements chosen independently at rate p. For $0 < \varepsilon < 1$, we have* $\Pr[\mathsf{Ham}_A(S,T) \in (1 \pm \varepsilon)p\mathsf{Ham}(S,T)] \geq 1 - 2\exp\left(-\frac{p\mathsf{Ham}(S,T)\varepsilon^2}{3}\right)$.

The triangle inequality yields $\mathsf{Ham}(S,U) \leq \mathsf{Ham}(S,T) + \mathsf{Ham}(T,U)$ for $S,T,U \in \Sigma^n$. The underlying phenomenon also allows retrieving $\mathsf{MI}(S,U)$ from $\mathsf{MI}(S,T)$ and $\mathsf{MI}(T,U)$. The following fact is proved in Appendix B.

▶ **Fact 3.2.** *For every $S,T,U \in \Sigma^n$ and every $A \subseteq [n]$, the mismatch information $\mathsf{MI}_A(S,U)$ can be retrieved from $\mathsf{MI}_A(S,T)$ and $\mathsf{MI}_A(T,U)$ in time $\tilde{O}(\mathsf{Ham}_A(S,T) + \mathsf{Ham}_A(T,U))$.*

Periods. An integer p is a *period* of $S \in \Sigma^*$ if and only if $S[i] = S[i+p]$ for all $1 \leq i \leq |S| - p$. The shortest period of S is denoted $\mathsf{per}(S)$. If $\mathsf{per}(S) \leq \frac{1}{2}|S|$, we say that S is *periodic*.

Rotations. For a string $S = S[1]S[2]\cdots S[n]$, let $\mathsf{cyc}(S) = S[2]\cdots S[n]S[1]$. For $i \in \mathbb{Z}$, we denote $i \circlearrowright n = ((i-1) \bmod n) + 1$ so that, for $i \in [n]$, the value $(i-1) \circlearrowright n$ is the position of $S[i]$ in $\mathsf{cyc}(S)$.[5] Moreover, for $M \subseteq \mathbb{Z}$, we denote $M \circlearrowright n = \{i \circlearrowright n \mid i \in M\}$. For $P \subseteq [n]$, let $\mathsf{rot}_n(P) = \{(i-1) \circlearrowright n \mid i \in P\}$ be the rotated set P.

The *primitive root* of a string S is the shortest string Q such that $S = Q^\alpha$ for an integer $\alpha \geq 1$. The length of the primitive root is denoted by $\mathsf{root}(S)$. Notice that $\mathsf{per}(S) \leq \mathsf{root}(S)$. Moreover, for every $m, m' \in \mathbb{Z}$, we have that $\mathsf{root}(\mathsf{cyc}^m(S)) = \mathsf{root}(S)$, and $\mathsf{cyc}^m(S) = \mathsf{cyc}^{m'}(S)$ if and only if $\mathsf{root}(S) \mid (m - m')$.

4 Sketches for Non-pseudo-periodic Strings

We say that a string $S \in \Sigma^n$ is (α,β)-*pseudo-periodic* if there exists a string $S' \in \Sigma^n$, called an (α,β)-*base* of S, such that $\mathsf{root}(S') \leq \frac{n}{\alpha}$ and $\mathsf{Ham}(S,S') \leq \beta$.

▶ **Observation 4.1.** *If S is (α,β)-pseudo-periodic with an (α,β)-base S', then every rotation $\mathsf{cyc}^m(S)$ with $m \in \mathbb{Z}$ is also (α,β)-pseudo-periodic and $\mathsf{cyc}^m(S')$ is an (α,β)-base of $\mathsf{cyc}^m(S)$.*

[5] We introduce the $\circlearrowright$ operator because positions in strings are indexed from 1 rather than from 0.

Let $\mathcal{H}_{n,k}$ be the set of strings in Σ^n that are $(3\gamma k, \gamma k)$-pseudo-periodic, where γ is the smallest constant such that $\gamma \geq 14$ and $\frac{n}{3\gamma k}$ is an integer. In this section, we present two circular sketches for strings in $\Sigma^n \setminus \mathcal{H}_{n,k}$: an (ε, k)-ACS sketch and a k-ECS sketch. Both sketches rely on the following result, proved in Section 5.

▶ **Theorem 4.2.** *For every two integers $1 \leq k \leq n$, there exists a randomized function $f : \Sigma^n \setminus \mathcal{H}_{n,k} \to 2^{[n]}$ such that the following holds for every $S_1, S_2 \in \Sigma^n \setminus \mathcal{H}_{n,k}$:*
1. *$|f(S_1)| = \tilde{O}(k)$ with high probability,*
2. *$f(\mathsf{cyc}(S_1)) = \mathsf{rot}_n(f(S_1))$,*
3. *if $\mathsf{Ham}(S_1, S_2) \leq k$, then $|f(S_1) \cap f(S_2)| \geq k$ with high probability.*

4.1 An (ε, k)-ACS Sketch

We start with briefly presenting a useful technical tool, that is, the non-circular version of the approximate sketch. We remark that many variants of this sketch exist, with equivalent space complexity. A short proof is given in Appendix B for the sake of completeness.

▶ **Theorem 4.3** ($(1 \pm \varepsilon)$-approximate sketches, folklore)**.** *There exists a $(1 \pm \varepsilon)$-approximate sketch $\mathtt{sk}_\varepsilon$ such that, given $\mathtt{sk}_\varepsilon(S_1)$ and $\mathtt{sk}_\varepsilon(S_2)$ for two strings $S_1, S_2 \in \Sigma^n$, one can decode $\mathsf{Ham}(S_1, S_2)$ with a $(1 \pm \varepsilon)$-multiplicative error. The sketches use $\tilde{O}(\varepsilon^{-2})$ space, the decoding algorithm is correct with high probability and costs $\tilde{O}(\varepsilon^{-2})$ time.*

Next, we describe our sketching scheme and prove that, together with an appropriate decoding algorithm, it forms an (ε, k)-ACS sketch for $\Sigma^n \setminus \mathcal{H}_{n,k}$.

▶ **Construction 4.4.** The encoding function $\mathtt{circ}_{\varepsilon,k} : \Sigma^n \setminus \mathcal{H}_{n,k} \to \{0,1\}^*$ is defined as follows:
1. Let $f : \Sigma^n \setminus \mathcal{H}_{n,k} \to 2^{[n]}$ be the selection function of Theorem 4.2.
2. Let $\mathtt{sk}_\varepsilon : \Sigma^n \to \{0,1\}^*$ be the sketch of Theorem 4.3.
3. Let $A, B \subseteq [n]$ be two subsets[6] with elements sampled independently with rate $p = 2\sqrt{\frac{\ln n}{k}}$.
4. For $S \in \Sigma^n \setminus \mathcal{H}_{n,k}$, the encoding $\mathtt{circ}_{\varepsilon,k}(S)$ stores $(i, \mathtt{sk}_\varepsilon(\mathsf{cyc}^i(S)))$ for $i \in f(S) \cap (A \cup B)$.

▶ **Proposition 4.5.** *There exists a decoding function which, together with the encoding $\mathtt{circ}_{\varepsilon,k}$ of Construction 4.4, forms an (ε, k)-ACS sketch of $\Sigma^n \setminus \mathcal{H}_{n,k}$. The size of this sketch is $\tilde{O}(\varepsilon^{-2}\sqrt{k})$, and the decoding algorithm costs $\tilde{O}(\sqrt{k} + \varepsilon^{-2})$ time with high probability.*

Proof. Our decoding procedure iterates over $i \in f(S_1) \cap A$. If $i' := (i + m) \circlearrowright n \in f(S_2) \cap B$, the procedure retrieves the sketches $\mathtt{sk}_\varepsilon(\mathsf{cyc}^i(S_1))$ and $\mathtt{sk}_\varepsilon(\mathsf{cyc}^{i'}(S_2))$ and recovers a $(1+\varepsilon)$-approximation of $\mathsf{Ham}(\mathsf{cyc}^i(S_1), \mathsf{cyc}^{i'}(S_2)) = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$. Otherwise, ∞ is returned.

We now reason that if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, then, with high probability, $i' \in f(S_2) \cap B$ for some $i \in f(S_1) \cap A$. By Theorem 4.2, $|f(S_1) \cap f(\mathsf{cyc}^m(S_2))| \geq k$. Thus, for any $i \in f(S_1) \cap f(\mathsf{cyc}^m(S_2))$, we have that $i \in f(S_1) \cap A$ with probability p. Similarly, $i' \in f(S_2) \cap B$ with probability p. Since A and B are independent, we have a success probability p^2 for each i independently. The probability of at least one success is at least $1 - (1-p^2)^k \geq 1 - n^{-4}$.

The decoding time is given by the time needed to compute the intersection of $f(S_1) \cap A$ and $\mathsf{rot}_n^m(f(S_2) \cap B)$, which is $\tilde{O}(\sqrt{k})$ with high probability, and $\tilde{O}(\varepsilon^{-2})$ time to decode the distance from a single pair of indices i, i', provided that the intersection is not empty. ◀

[6] The sketch would remain valid with one subset only. However, introducing the second subset simplifies the arguments and makes the construction more similar to the counterpart for pseudo-periodic strings.

4.2 An k-ECS Sketch

We begin with the following corollary of [37, Theorem 5.1]. The original statement in [37] is given for $A = [n]$ only, but it can be generalized in a straightforward manner, e.g., by replacing all characters at positions in $[n] \setminus A$ with a fixed character.

▶ **Theorem 4.6** (based on [37, Theorem 5.1]). *For every $k \leq n$ and $A \subseteq [n]$, there is a sketch $\mathtt{sk}_{k,A}$ of size $\tilde{O}(k)$ such that, given $\mathtt{sk}_{k,A}(S_1)$ and $\mathtt{sk}_{k,A}(S_2)$ for two strings $S_1, S_2 \in \Sigma^n$:*

- *if $\mathsf{Ham}_A(S_1, S_2) \leq k$, then the decoding function returns $\mathsf{MI}_A(S_1, S_2)$;*
- *otherwise, if $\mathsf{Ham}_A(S_1, S_2) > k$, the decoding function reports that this is the case.*

The decoding algorithm is correct with high probability and costs $\tilde{O}(k)$ time.

▶ **Construction 4.7.** The encoding function $\mathtt{circ}_k : \Sigma^n \setminus \mathcal{H}_{n,k} \to \{0,1\}^*$ is defined as follows:

1. Let $f : \Sigma^n \setminus \mathcal{H}_{n,k} \to 2^{[n]}$ be the selection function of Theorem 4.2.
2. Let $A \subseteq [n]$ be a subset with elements sampled independently with rate $p := \frac{9 \ln n}{k}$.
3. Denote $t = \lceil 18 \ln n \rceil$, and let $\mathtt{sk}_{t,A} : \Sigma^n \to \{0,1\}^*$ be the sketch of Theorem 4.6.
4. For $S \in \Sigma^n \setminus \mathcal{H}_{n,k}$, the encoding $\mathtt{circ}_k(S)$ stores the pairs $(i, \mathtt{sk}_{t,A}(\mathsf{cyc}^i(S)))$ for $i \in f(S)$.

▶ **Proposition 4.8.** *There exists a decoding function which, together with the encoding $\mathtt{circ}_k$ of Construction 4.7, forms a k-ECS sketch of $\Sigma^n \setminus \mathcal{H}_{n,k}$. The size of this sketch is $\tilde{O}(k)$, and the decoding algorithm costs $\tilde{O}(k)$ time with high probability.*

Proof. The decoding procedure iterates over $i \in f(S_1) \cap \mathsf{rot}_n^m(f(S_2))$, If the number of such positions is less than k, then ∞ is returned. Otherwise, for each $i \in f(S_1) \cap \mathsf{rot}_n^m(f(S_2))$, we have $i' := (i + m) \circlearrowright n \in f(S_2)$, and the algorithm runs a decoding procedure for $\mathtt{sk}_{t,A}(\mathsf{cyc}^i(S_1))$ and $\mathtt{sk}_{t,A}(\mathsf{cyc}^{i'}(S_2))$. If any such decoding fails, then ∞ is returned. Otherwise, for each mismatch position j found, say with $\mathsf{cyc}^i(S_1)[j] \neq \mathsf{cyc}^{i'}(S_2)[j]$, the algorithm adds $(i + j) \circlearrowright n$ to a set M, initialized as the empty set. Finally, the size $|M|$ is returned.

The decoding procedure costs $\tilde{O}(k)$ time, which is needed both to find all the aligned pairs $i \in f(S), i' \in f(S_2)$ by computing the intersection $f(S_1) \cap \mathsf{rot}_n^m(f(S_2))$ and to retrieve and gather the mismatches obtained from the aligned pairs (in $\tilde{O}(t) = \tilde{O}(1)$ time per pair).

Correctness. Recall that $\mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$ is the set of mismatch positions between S_1 and $\mathsf{cyc}^m(S_2)$. First, notice that each $j \in M$ is a mismatch position between S_1 and $\mathsf{cyc}^m(S_2)$, since $\mathsf{cyc}^i(S_1)[j] \neq \mathsf{cyc}^{i'}(S_2)[j]$ is equivalent to $S_1[(i + j) \circlearrowright n] \neq S_2[(i + j + m) \circlearrowright n]$. Hence, $M \subseteq \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$ and $|M| \leq \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$

Now, we prove that, with high probability, if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, then the algorithm reports $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$, and if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) > k$, then the algorithm reports a value larger than k. In the case where $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, we have that $|f(S_1) \cap \mathsf{rot}_n^m(f(S_2))| \geq k$ with high probability due to Theorem 4.2. Moreover, for every $i \in f(S_1) \cap \mathsf{rot}_n^m(f(S_2))$, the expected number of positions in $A \cap \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$ is $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \cdot p \leq k \cdot \frac{9 \ln n}{k} = 9 \ln n$. Hence, by a Chernoff bound $\Pr[|A \cap \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))| > 18 \ln n] \leq \exp(-\frac{9 \ln n}{3}) = n^{-3}$. Thus, when $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, decoding $\mathtt{sk}_{t,A}(\mathsf{cyc}^i(S_1))$ and $\mathtt{sk}_{t,A}(\mathsf{cyc}^{i'}(S_2))$ succeeds for all $i \in f(S_1) \cap \mathsf{rot}_n^m(f(S_2))$ with high probability.

Conditioned on the event that $|f(S_1) \cap \mathsf{rot}_n^m(f(S_2))| \geq k$ and the decoding algorithm of $\mathtt{sk}_{t,A}$ is successful, we now prove that $|M| = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$. For each mismatch $j \in \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$, there is an independent trial associated with each $i \in f(S_1) \cap \mathsf{rot}_n^m(f(S_2))$, which is whether $((j - i) \circlearrowright n) \in A$ or not. The trial is successful with probability p. The probability that at least one of those trials succeeds is at least $1 - (1-p)^k \geq 1 - n^{-9}$. Applying the union bound over all $j \in \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$, we conclude that $M = \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$ and $|M| = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ with high probability.

If $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) > k$, then the decoding algorithm may return ∞ because of $|f(S_1) \cap \mathsf{rot}_n^m(f(S_2))| < k$ or due to a decoding failure. If neither of these events happen, the algorithm returns $|M|$, which is equal to $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ with high probability (as proved above). Thus, in both cases, a value larger than k is reported. ◀

5 Construction of the Selection Function

For $S \in \Sigma^n$, let $S^* = S \cdot S \cdot S \cdots$ be the infinite string which is the infinite concatenation of S to itself (for any $i \in \mathbb{N}$, we have $S^*[i] = S[i \circlearrowright n]$). Let $\ell = \frac{n}{3\gamma k}$ (recall it is an integer). A position $i \in [n]$ is called *cubic* if $u_i = S^*[i \,..\, i+3\ell-1]$ has $\mathsf{per}(u_i) \le \frac{|u_i|}{3} = \ell$, i.e., if the cyclic fragment of length 3ℓ starting at position i consists of at least three repetitions of the same factor. Otherwise, position i is called *non-cubic*. We denote the set of cubic positions in a string S as $\mathsf{C}(S)$, and the set of non-cubic positions as $\mathsf{N}(S)$. Notice that $\mathsf{C}(S) \cup \mathsf{N}(S) = [n]$ and $\mathsf{C}(S) \cap \mathsf{N}(S) = \emptyset$.

We present two selection techniques, resulting in functions f_{n} and f_{c}, designed for strings with many non-cubic positions and for strings with many cubic positions, respectively. Both functions satisfy the first two properties of Theorem 4.2 for any string $S_1 \in \Sigma^n \setminus \mathcal{H}_{n,k}$. The functions f_{n} and f_{c} have the third property of Theorem 4.2 if $|\mathsf{N}(S_1)| \ge \frac{n}{2}$ and if $|\mathsf{C}(S_1)| \ge \frac{n}{2}$, respectively. Thus, the function f defined through $f(S) = f_{\mathsf{n}}(S) \cup f_{\mathsf{c}}(S)$ satisfies Theorem 4.2.

5.1 Selecting Positions for Strings with Many Non-cubic Positions

Throughout this subsection, let $h : \Sigma^{3\ell} \to \{0,1\}$ be a hash function assigning values independently to each $u \in \Sigma^{3\ell}$ such that $\Pr[h(u) = 1] = \frac{4k \ln n}{n}$. For clarity, we omit the explicit dependence on h in our notation. For $S \in \Sigma^n$, define $f_{\mathsf{n}}(S) = \{i \in \mathsf{N}(S) \mid h(u_i) = 1\}$.

Our proofs rely on the following multiplicative Chernoff–Hoeffding bound:

▶ **Proposition 5.1** (Corollary of [20, Theorems 1.10.1 and 1.10.5]). *Let $X_1, \ldots, X_n$ be independent random variables taking values in $[0, M]$, let $X = \sum_{i=1}^n X_i$, and let $\mu \ge 0$.*

(a) *If $\mu \ge \mathbb{E}[X]$, then, for every $\delta > 0$, we have $\Pr[X \ge (1+\delta)\mu] \le \exp(-\frac{\min(\delta,\delta^2)\mu}{3M})$.*

(b) *If $\mu \le \mathbb{E}[X]$, then, for every $0 < \delta < 1$, we have $\Pr[X \le (1-\delta)\mu] \le \exp(-\frac{\delta^2\mu}{2M})$.*

We first prove that f_{n} satisfies the first property of Theorem 4.2.

▶ **Lemma 5.2.** *For every $S \in \Sigma^n$, we have $\Pr\left[|f_{\mathsf{n}}(S)| < 8k \ln n\right] \ge 1 - n^{-\Omega(1)}$.*

Proof. For each $u \in \Sigma^{3\ell}$, we introduce a random variable $X_u = |\{i \in f_{\mathsf{n}}(S) \mid u_i = u\}|$; notice that X_u depends only on $h(u)$, so the variables X_u are independent. In order to apply Proposition 5.1 for $|f_{\mathsf{n}}(S)| = \sum_{u \in \Sigma^{3\ell}} X_u$, we prove that each X_u is bounded.

First, note that if $\mathsf{per}(u) \le \ell$ or $h(u) = 0$, then $X_u = 0$. Otherwise, as $u_i = u = u_{i'}$ for $i < i' \le i + 3\ell$ implies $i' - i \ge \mathsf{per}(u) > \ell$, we conclude that $X_u = |\{i \in [n] \mid u_i = u\}| \le \frac{n}{\ell} = 3\gamma k$. Now, $\mathbb{E}[|f_{\mathsf{n}}(S)|] = \sum_{i \in \mathsf{N}(S)} \Pr[h(u_i) = 1] = |\mathsf{N}(S)| \cdot \frac{4k \ln n}{n} \le 4k \ln n$, so, by Proposition 5.1(a) with $\delta = 1$, we have $\Pr[|f_{\mathsf{n}}(S)| \ge 8k \ln n] \le \exp(-\frac{4k \ln n}{3 \cdot 3\gamma k}) = n^{-4/(9\gamma)} = n^{-\Omega(1)}$. ◀

The following lemma states that f_{n} satisfies Property 2 of Theorem 4.2.

▶ **Lemma 5.3.** *For every $S \in \Sigma^n$, we have $f_{\mathsf{n}}(\mathsf{cyc}(S)) = \mathsf{rot}_n(f_{\mathsf{n}}(S))$.*

Proof. Let $i \in f_{\mathsf{n}}(\mathsf{cyc}(S))$ and let $u = (\mathsf{cyc}(S))^*[i \,..\, i + \ell - 1] = S^*[i+1 \,..\, i+\ell]$. Since $i \in f_{\mathsf{n}}(\mathsf{cyc}(S))$, we have that $\mathsf{per}(u) > \frac{\ell}{3}$ and $h(u) = 1$. Therefore, $(i+1) \circlearrowright n \in f_{\mathsf{n}}(S)$, which means that $i \circlearrowright n = i \in \mathsf{rot}_n(f_{\mathsf{n}}(S))$. Hence, $f_{\mathsf{n}}(\mathsf{cyc}(S)) \subseteq \mathsf{rot}_n(f_{\mathsf{n}}(S))$. Symmetrically, $\mathsf{rot}_n(f_{\mathsf{n}}(S)) \subseteq f_{\mathsf{n}}(\mathsf{cyc}(S))$. Thus, $f_{\mathsf{n}}(\mathsf{cyc}(S)) = \mathsf{rot}_n(f_{\mathsf{n}}(S))$. ◀

Finally, the following lemma states that f_{n} satisfies Property 3 of Theorem 4.2.

▶ **Lemma 5.4.** *Suppose that* $S_1, S_2 \in \Sigma^n$ *satisfy* $\mathsf{Ham}(S_1, S_2) \leq k$. *If* $|\mathsf{N}(S_1)| \geq \frac{1}{2}n$, *then* $\Pr[|f_{\mathsf{n}}(S_1) \cap f_{\mathsf{n}}(S_2)| \geq k] \geq 1 - n^{-\Omega(1)}$.

Proof. For each $i \in [n]$, let $u_i = S_1^*[i\,..\,i+3\ell-1]$ and $v_i = S_2^*[i\,..\,i+3\ell-1]$, and let $\Lambda = \{i \in \mathsf{N}(S_i) \mid u_i = v_i\}$. Notice that, for $i \in [n]$, we have $u_i \neq v_i$ if and only if $\mathsf{MP}(S_1, S_2) \cap ([i\,..\,i+3\ell-1] \circlearrowleft n) \neq \emptyset$. Hence, the number of indices $i \in [n]$ with $u_i \neq v_i$ is at most $|\mathsf{MP}(S_1, S_2)| \cdot 3\ell \leq k \cdot \frac{n}{\gamma k} \leq \frac{n}{\gamma}$. Since $|\mathsf{N}(S_1)| \geq \frac{1}{2}n$, then $|\Lambda| \geq \frac{1}{2}n - \frac{1}{\gamma}n > \frac{1}{3}n$ due to $\gamma \geq 6$. Thus, $\mathbb{E}[|f_{\mathsf{n}}(S_1) \cap f_{\mathsf{n}}(S_2)|] \geq |\Lambda| \cdot \frac{4k \ln n}{n} \geq \frac{4}{3}k \ln n$. The rest of the proof follows from Proposition 5.1(b) similarly as Proposition 5.1(a) is applied in the proof of Lemma 5.2. ◀

5.2 Selecting Positions for Strings with Many Cubic Positions

Recall that our goal is to design a rotation-invariant mechanism for selecting $\tilde{O}(k)$ indices so that, given two fairly similar strings, at least k common indices are selected in both strings. In the selection procedure described in Section 5.1, the decision whether or not to include position i was based on whether or not $S^*[i\,..\,i+3\ell-1] \in \Pi$ for a certain family $\Pi \subseteq \Sigma^{3\ell}$. Then, we argued that $S_1^*[i\,..\,i+3\ell-1] = S_2^*[i\,..\,i+3\ell-1] \in \Pi$ for at least k positions $i \in [n]$.

Unfortunately, this strategy might be infeasible if $\mathsf{C}(S)$ is large, that is, when there is a large number of cubic positions in S. For example, it could be the case that $S_1^*[i\,..\,i+3\ell-1] \neq S_2^*[i\,..\,i+3\ell-1]$ holds for $3\ell k = \frac{n}{\gamma}$ positions $i \in [n]$, and $S_1^*[i\,..\,i+3\ell-1] = S_2^*[i\,..\,i+3\ell-1] = \mathsf{a}^{3\ell}$ for the remaining $n - \frac{n}{\gamma}$ positions $i \in [n]$. This may happen even if $\mathsf{Ham}(S_1, \mathsf{a}^n) = \Omega(\frac{n}{\gamma})$, i.e., for strings far from being $(3\gamma k, \gamma k)$-pseudo-periodic.

We begin with some intuition for the construction of the function f_{c}. First, suppose that, for each position $i \in \mathsf{C}(S)$, we include in $f_{\mathsf{c}}(S)$ the smallest $j > i$ such that $\mathsf{per}(S^*[i\,..\,j]) > \mathsf{per}(S^*[i\,..\,i+3\ell-1])$. In other words, $f_{\mathsf{c}}(S)$ contains the positions following each maximal cyclic fragment of length at least 3ℓ and period at most ℓ. Notice that this construction satisfies Property 2 of Theorem 4.2. Moreover, since each position may belong to at most two such maximal repetitions, the number of positions selected is at most $\frac{2n}{3\ell} = 2\gamma k$ (so that Property 1 of Theorem 4.2 is satisfied), and a substitution of a single character in S may remove at most two positions from $f_{\mathsf{c}}(S)$. However, if the cubic positions are clustered in few blocks, then this mechanism is not enough to guarantee that Property 3 of Theorem 4.2 is satisfied, i.e., that $|f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)| \geq k$ when $\mathsf{Ham}(S_1, S_2) \leq k$. Hence, instead of selecting just one position j for each $i \in \mathsf{C}(S)$, several positions are selected using a process inspired by [10] with subsequent improvements in [13]: The fragment $S^*[i\,..\,i+3\ell-1]$ is maximally extended to $S^*[i\,..\,i+\tau_i-1]$ so that the period of $S^*[i\,..\,i+\tau_i-1]$ drops to $\mathsf{per}(S^*[i\,..\,i+3\ell-1])$ after $\Theta(\frac{k}{n}\tau_i)$ substitutions, and the underlying mismatching positions are added to $f_{\mathsf{c}}(S)$.

5.2.1 Definition of f_{c}

For any $i \in \mathsf{C}(S^*)$, let $u_i = S^*[i\,..\,i+3\ell-1]$, let $\rho_i = \mathsf{per}(u_i)$, and let $\mu_{S,i} = S^*[i\,..\,i+\rho_i-1]$, which is the string period of u_i. To avoid clutter in the presentation, we use $\mu_i = \mu_{S,i}$ when S is clear from context. Notice that, for $\tau \geq 2\rho_i$, the string $\mu_i^*[1\,..\,\tau]$ is the (unique) string of length τ with string period μ_i.

We are now ready to formally define the concept of extending (to the right) a cubic fragment starting at position i for as long as the ratio between the length of the extended fragment and the Hamming distance between the extended fragment and the appropriate prefix of μ_i^* is large enough. The length of such a (maximal) extended fragment is defined as

$$\tau_{S,i} = \min\left\{\tau \mid \tau < \tfrac{n}{\gamma k}\mathsf{Ham}\left(S^*[i\,..\,i+\tau-1], \mu_i^*[1\,..\,\tau]\right)\right\}.$$

The following lemma shows that $\tau_{S,i}$ is well-defined, i.e., that the minimum in the definition of $\tau_{S,i}$ is taken over a non-empty set. The bound $\tau_{S,i} \le 2n$ is also useful later on.

▶ **Lemma 5.5.** *For every $S \in \Sigma^n \setminus \mathcal{H}_{n,k}$ and $i \in \mathsf{C}(S)$, we have $\tau_{S,i} \le 2n$.*

Proof. Let $i \in \mathsf{C}(S)$ and assume by contradiction that $\tau_{S,i} > 2n$. This yields

$$2n \ge \tfrac{n}{\gamma k}\mathsf{Ham}\left(S^*[i\,..\,i+2n-1], \mu_i^*[1\,..\,2n]\right).$$

Moreover, $S^*[i\,..\,i+n-1] = S^*[i+n\,..\,i+2n-1]$, and so, by the triangle inequality,

$$\begin{aligned}
2\gamma k &\ge \mathsf{Ham}\left(S^*[i\,..\,i+2n-1], \mu_i^*[1\,..\,2n]\right)\\
&= \mathsf{Ham}\left(S^*[i\,..\,i+n-1], \mu_i^*[1\,..\,n]\right) + \mathsf{Ham}\left(S^*[i+n\,..\,i+2n-1], \mu_i^*[n+1\,..\,2n]\right)\\
&= \mathsf{Ham}\left(S^*[i\,..\,i+n-1], \mu_i^*[1\,..\,n]\right) + \mathsf{Ham}\left(S^*[i\,..\,i+n-1], \mu_i^*[n+1\,..\,2n]\right)\\
&\ge \mathsf{Ham}\left(\mu_i^*[1\,..\,n], \mu_i^*[n+1\,..\,2n]\right).
\end{aligned}$$

Notice that for any strings x, y, z (with $|x| = |y|$) and any integer m, we have $\mathsf{Ham}(x,y) = \frac{1}{m}\mathsf{Ham}(x^m, y^m)$ and $\mathsf{Ham}(x,y) \le \mathsf{Ham}(xz, yz)$. Thus, due to $|\mu_i| = \rho_i \le \ell \le \frac{n}{3\gamma k}$, we have

$$\begin{aligned}
\mathsf{Ham}\left(\mu_i, \mu_i^*[n+1\,..\,n+\rho_i]\right) &= \tfrac{1}{3\gamma k}\mathsf{Ham}\left(\mu_i^*[1\,..\,3\gamma k\rho_i], \mu_i^*[n+1\,..\,n+3\gamma k\rho_i]\right)\\
&\le \tfrac{1}{3\gamma k}\mathsf{Ham}\left(\mu_i^*[1\,..\,n], \mu_i^*[n+1\,..\,2n]\right) \le \tfrac{2\gamma k}{3\gamma k} < 1.
\end{aligned}$$

Consequently, $\mu_i = \mu_i^*[n+1\,..\,n+\rho_i] = \mathsf{cyc}^n(\mu_i)$, which implies $\rho_i \mid n$ by primitivity of μ_i (recall that $\mu_i = \mathsf{cyc}^m(\mu_i)$ only for $\rho_i \mid m$). Since $\tau_{S,i} > n$, we have $n \ge \frac{n}{\gamma k}\mathsf{Ham}(S^*[i\,..\,i+n-1], \mu_i^*[1\,..\,n])$, that is $\gamma k \ge \mathsf{Ham}\big(S^*[i\,..\,i+n-1], \mu_i^*[1\,..\,n]\big) = \mathsf{Ham}\big(S^*[i\,..\,i+n-1], \mu_i^{n/\rho_i}\big)$. Hence, $S^*[i\,..\,i+n-1] \in \mathcal{H}_{n,k}$ so, by Observation 4.1, $S \in \mathcal{H}_{n,k}$. ◀

Let $R_{S,i} = [i\,..\,i+\tau_i-1]$ be the positions in the extended fragment, and let $M_{S,i} = \{j \in R_{S,i} \mid S[j] \ne \mu_i^*[j-i+1]\}$ be the set of positions in $R_{S,i}$ corresponding to mismatches between $S^*[i\,..\,i+\tau_i-1]$ and $\mu_i^*[1\,..\,\tau_i]$. To avoid clutter in the presentation, we use $\tau_i = \tau_{S,i}$, $R_i = R_{S,i}$, and $M_i = M_{S,i}$ when S is clear from context. Define

$$f_{\mathsf{c}}(S) = \bigcup_{i \in \mathsf{C}(S)} (M_i \circlearrowleft n) = \{p \circlearrowleft n \mid p \in M_i, i \in \mathsf{C}(S)\}.$$

5.2.2 Properties of f_{c}

Property 1 of Theorem 4.2. Our strategy for proving an upper bound on the size of $f_{\mathsf{c}}(S)$ is to associate each $i \in \mathsf{C}(S)$ with a carefully defined set $A_i \subseteq R_i$. We then select a subset $\Gamma \subseteq \mathsf{C}(S)$ so that the sets A_i for $i \in \Gamma$ are disjoint subsets of $[1\,..\,3n]$ and $\bigcup_{i\in\Gamma} M_i = \bigcup_{i\in\mathsf{C}(S)} M_i$. Finally, we show that $|M_i| = O(\frac{\gamma k}{n}|A_i|)$ for each $i \in \mathsf{C}(S)$, and so $|\bigcup_{i\in\mathsf{C}(S)} M_i| = |\bigcup_{i\in\Gamma} M_i| = O(\sum_{i\in\Gamma}\frac{\gamma k}{n}|A_i|) = O(\gamma k)$.

For each R_i, consider the set of indices $j \in R_i$ such that $[j, j+2\ell) \cap M_i = \emptyset$. Formally, let $A_i = \{j \in R_i \mid [j, j+2\ell) \subseteq R_i \setminus M_i\}$. The following lemma lets us define $f_{\mathsf{c}}(S)$ as the union of $M_i \circlearrowleft n$ for a restricted set of values of i, with the property of having disjoint sets A_i.

▶ **Lemma 5.6.** *Let $i, i' \in \mathsf{C}(S)$. If $i < i'$ and $A_i \cap A_{i'} \ne \emptyset$, then $M_{i'} \subseteq M_i$.*

The following fact is useful in the proof of Lemma 5.6.

▶ **Fact 5.7** ([24, Lemma 6])**.** *Let S be a periodic string. If T is a substring of S of length at least $2\mathsf{per}(S)$, then $\mathsf{per}(S) = \mathsf{per}(T)$.*

Proof of Lemma 5.6. Let $j \in A_i \cap A_{i'}$. By definition, $[j\,..\,j+2\ell) \subseteq (R_i \setminus M_i) \cap (R_{i'} \setminus M_{i'})$. Thus, $\mu_i^*[1{+}j{-}i\,..\,2\ell{+}j{-}i] = S^*[j\,..\,j{+}2\ell{-}1] = \mu_{i'}^*[1{+}j{-}i'\,..\,2\ell{+}j{-}i']$. Since $\rho_i = \mathsf{per}(\mu_i^*) \leq \ell$ and $\rho_{i'} = \mathsf{per}(\mu_{i'}^*) \leq \ell$, by Fact 5.7, we have $\rho_i = \mathsf{per}(\mu_i^*) = \mathsf{per}(\mu_i^*[1+j-i\,..\,2\ell+j-i]) = \mathsf{per}(\mu_{i'}^*[1+j-i'\,..\,2\ell+j-i']) = \rho_{i'}$. Therefore, $\mu_{i'}^*[1\,..\,\tau_{i'}] = \mu_i^*[i'-i+1\,..\,i'-i+\tau_{i'}]$ (since the two fragments are extensions of the same periodic string with the same period). Hence, for any $\tau \leq \tau_{i'}$, we have $\mathsf{Ham}(S^*[i'\,..\,i'+\tau-1], \mu_{i'}^*[1\,..\,\tau]) = \mathsf{Ham}(S^*[i'\,..\,i'+\tau-1], \mu_i^*[i'-i+1\,..\,i'-i+\tau])$.

Since $\min(A_i \cap A_{i'}) \geq i'$ and $A_i \subseteq R_i$, we have that $\tau_i > i'-i$. Therefore, for $\tau = i'-i$, we have $i'-i \geq \frac{n}{\gamma k}\mathsf{Ham}(S^*[i\,..\,i+i'-i-1], \mu_i^*[1\,..\,i'-i]) = \frac{n}{\gamma k}\mathsf{Ham}(S^*[i\,..\,i'-1], \mu_i^*[1\,..\,i'-i])$.

Thus, for any $\tau < i'-i+\tau_{i'}$, we have

$$\begin{aligned}
&\tfrac{n}{\gamma k}\mathsf{Ham}\left(S^*[i\,..\,i+\tau-1], \mu_i^*[1\,..\,\tau]\right)\\
&\quad= \tfrac{n}{\gamma k}\mathsf{Ham}\left(S^*[i\,..\,i'-1], \mu_i^*[1\,..\,i'-i]\right) + \tfrac{n}{\gamma k}\mathsf{Ham}\left(S^*[i'\,..\,i+\tau-1], \mu_i^*[i'-i+1\,..\,\tau]\right)\\
&\quad\leq i'-i+\tfrac{n}{\gamma k}\mathsf{Ham}\left(S^*[i'\,..\,i'-(i'-i)+\tau-1], \mu_{i'}^*[1\,..\,\tau-(i'-i)]\right)\\
&\quad\leq i'-i+\tau-(i'-i) = \tau.
\end{aligned}$$

Consequently, $\tau_i \geq i'-i+\tau_{i'}$, which means that $R_{i'} \subseteq R_i$. For a proof that $M_{i'} \subseteq M_i$, let us choose $j' \in M_{i'}$. By definition, $S[j'] \neq \mu_{i'}^*[j'-i'+1] = \mu_i^*[j'-i'+1+(i'-i)] = \mu_i^*[j'-i+1]$. Hence, $j' \in M_i$. ◀

Lemma 5.6 implies that for any two indices $i < i'$, if $A_i \cap A_{i'} \neq \emptyset$, then $M_{i'} \subseteq M_i$, and thus it is enough to consider only the index i when defining $f_{\mathsf{c}}(S)$. Therefore, we define $\Gamma = \{i' \in \mathsf{C}(S) \mid \forall i < i' : A_i \cap A_{i'} = \emptyset\}$. Notice that, among $i \in \Gamma$, all the sets A_i are disjoint. Moreover, since for any $i \in \mathsf{C}(S)$ we have $A_i \subseteq R_i \subseteq [1\,..\,3n]$ by Lemma 5.5, we have $\sum_{i\in\Gamma}|A_i| = \left|\bigcup_{i\in\Gamma} A_i\right| \leq |[1\,..\,3n]| = 3n$.

For every $i \in \mathsf{C}(S)$, we have $|A_i| \geq |R_i| - 2\ell|M_i| = |R_i| - \frac{2n}{3\gamma k}|M_i|$. Furthermore, $|R_i|-1 \geq \frac{n}{\gamma k}(|M_i|-1)$ by definition of $\tau_i = |R_i|$. Thus, $|A_i| > \frac{n}{\gamma k}|M_i| - \frac{n}{\gamma k} - \frac{2n}{3\gamma k}|M_i| = \frac{n}{3\gamma k}(|M_i|-3)$. Due to $[i\,..\,i+\ell) \subseteq A_i$, we have $|A_i| \geq \ell = \frac{n}{3\gamma k}$, and therefore $|M_i| < \frac{3\gamma k}{n}|A_i| + 3 \leq \frac{3\gamma k}{n}|A_i| + \frac{9\gamma k}{n}|A_i| = \frac{12\gamma k}{n}|A_i|$. Hence, $|f_{\mathsf{c}}(S)| \leq \left|\bigcup_{i\in\mathsf{C}(S)} M_i\right| = \left|\bigcup_{i\in\Gamma} M_i\right| \leq \sum_{i\in\Gamma}|M_i| \leq \sum_{i\in\Gamma}\frac{12\gamma k}{n}|A_i| = \frac{12\gamma k}{n}\sum_{i\in\Gamma}|A_i| \leq 36\gamma k$.

Property 2 of Theorem 4.2. The following lemma states that f_{c} satisfies Property 2.

▶ **Lemma 5.8.** *For every $S \in \Sigma^n$, we have $f_{\mathsf{c}}(\mathsf{cyc}(S)) = \mathsf{rot}_n(f_{\mathsf{c}}(S))$.*

Proof. Let $j \in f_{\mathsf{c}}(\mathsf{cyc}(S))$. There exists $i \in \mathsf{C}(\mathsf{cyc}(S))$ such that $j \in M_{\mathsf{cyc}(S),i} \circlearrowright n$. Let $j' \in M_{\mathsf{cyc}(S),i}$ such that $j = j' \circlearrowright n$. We distinguish between two cases: if $i \in [1\,..\,n-1]$, then, since $i \in \mathsf{C}(\mathsf{cyc}(S))$, we have $i+1 \in \mathsf{C}(S)$ and $\tau_{S,i+1} = \tau_{\mathsf{cyc}(S),i}$. Therefore, $j'+1 \in M_{S,i+1}$ and $(j'+1) \circlearrowright n \in f_{\mathsf{c}}(S)$. Thus, $j = (j'+1-1) \circlearrowright n \in \mathsf{rot}_n(f_{\mathsf{c}}(S))$. If $i = n$, then it must be that $1 \in \mathsf{C}(S)$ and $\tau_{S,1} = \tau_{\mathsf{cyc}(S),n}$. Therefore, $j'-n+1 \in M_{S,1}$ and $(j'-n+1) \circlearrowright n \in f_{\mathsf{c}}(S)$. Thus, $j = (j'-n+1-1) \circlearrowright n \in \mathsf{rot}_n(f_{\mathsf{c}}(S))$. The converse inclusion holds symmetrically. ◀

Property 3 of Theorem 4.2. We first give a lower bound on $|f_{\mathsf{c}}(S)|$ in terms of $|\mathsf{C}(S)|$.

▶ **Lemma 5.9.** *For every string $S \in \Sigma^n \setminus \mathcal{H}_{n,k}$, we have $|f_{\mathsf{c}}(S)| \geq \frac{\gamma k}{3n}|\mathsf{C}(S)|$.*

Proof. First, we shall construct a set $\Delta \subseteq \mathsf{C}(S)$ such that $\sum_{i\in\Delta}|R_i| \geq |\mathsf{C}(S)|$ and, for any two distinct indices $i, i' \in \Delta$, we have $R_i \cap R_{i'} = \emptyset$. We build Δ iteratively. We start with $\Delta = \emptyset$ and, as long as $\mathsf{C}(S) \not\subseteq \bigcup_{i\in\Delta} R_i$, we add $\min\left(\mathsf{C}(S) \setminus \bigcup_{i\in\Delta} R_i\right)$ to Δ. Let $i < i'$ be

two indices in Δ. When i' was added to Δ, we already had $i \in \Delta$. Thus, R_i ends to the left of i', which is the starting point of $R_{i'}$. Hence, $R_i \cap R_{i'} = \emptyset$. The algorithm terminates when $\mathsf{C}(S) \subseteq \bigcup_{i\in\Delta} R_i$, so $|\mathsf{C}(S)| \le |\bigcup_{i\in\Delta} R_i| = \sum_{i\in\Delta} |R_i|$.

For any $i \in \mathsf{C}(S)$, we have $|R_i| = \tau_i < \frac{n}{\gamma k}\mathsf{Ham}\,(S^*[i\,..\,i+\tau_i-1], \mu_i^*[1\,..\,\tau_i])$, i.e., $|R_i| < \frac{n}{\gamma k}|M_i|$. Since $M_i \subseteq R_i$ for every i, the sets M_i for $i \in \Delta$ are disjoint. Consequently, $|\bigcup_{i\in\Delta} M_i| = \sum_{i\in\Delta} |M_i| > \frac{\gamma k}{n}\sum_{i\in\Delta} |R_i| \ge \frac{\gamma k}{n}|\mathsf{C}(S)|$.

By Lemma 5.5, for any $i \in \mathsf{C}(S)$, we have $\tau_i \le 2n$. Therefore, $\bigcup_{i\in\Delta} M_i \subseteq [1\,..\,3n]$ and each position in $j \in \bigcup_{i\in\Delta}(M_i \circlearrowleft n)$ may be introduced by at most 3 positions $j, j+n, j+2n \in \bigcup_{i\in\Delta} M_i$. Thus, $|f_{\mathsf{c}}(S)| = \left|\bigcup_{i\in\Delta}(M_i \circlearrowleft n)\right| \ge \frac{1}{3}\left|\bigcup_{i\in\Delta} M_i\right| \ge \frac{\gamma k}{3n}|\mathsf{C}(S)|$. ◀

Using Lemma 5.9, we prove the third property of Theorem 4.2, assuming $|\mathsf{C}(S_1)| \ge \frac{1}{2}n$.

▶ **Lemma 5.10.** *Suppose that $S_1, S_2 \in \Sigma^n \setminus \mathcal{H}_{n,k}$ satisfy $\mathsf{Ham}(S_1, S_2) \le k$. If $|\mathsf{C}(S_1)| \ge \frac{1}{2}n$, then $|f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)| \ge k$.*

Proof. Let S' be a string of length n, where, for any i with $S_1[i] = S_2[i]$, we have $S'[i] = S_1[i]$ and, for any other i (i.e., for $i \in \mathsf{MP}(S_1, S_2)$), we have $S'[i] = \$_i$, where $\$_i \notin \Sigma$ differs from any other character $\$_{i'}$ for $i' \ne i$.

▷ Claim 5.11. $f_{\mathsf{c}}(S') \subseteq (f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)) \cup \mathsf{MP}(S_1, S_2)$.

Proof. Let $j \in f_{\mathsf{c}}(S')$. If $j \in \mathsf{MP}(S_1, S_2)$, the claim follows; thus, assume $j \notin \mathsf{MP}(S_1, S_2)$. By the definition of $f_{\mathsf{c}}(S')$, there is an index $i \in \mathsf{C}(S')$ such that $j \in M_{S',i} \circlearrowleft n$; let $j' \in M_{S',i}$ be an integer such that $j = j' \circlearrowleft n$. Notice that $\mu_{S_1,i} = \mu_{S',i}$ since if $\mu_{S',i}$ contains some $\$_k$ character, then i cannot be cubic and so $i \notin \mathsf{C}(S')$. Therefore, $\mu_{S_1,i} = \mu_{S',i}$, and let $\mu_i = \mu_{S_1,i}$. For any integer τ, we have $\mathsf{Ham}(S_1^*[i\,..\,i+\tau-1], \mu_i^*[1\,..\,\tau]) \le \mathsf{Ham}((S')^*[i\,..\,i+\tau-1], \mu_i^*[1\,..\,\tau])$ because the new $\$_k$ characters in S' just form new mismatches. In particular, for $\tau_{S_1,i}$ we have $\frac{n}{\gamma k}\mathsf{Ham}((S')^*[i\,..\,i+\tau_{S_1,i}-1], \mu_i^*[1\,..\,\tau_{S_1,i}]) \ge \frac{n}{\gamma k}\mathsf{Ham}(S_1^*[i\,..\,i+\tau_{S_1,i}-1], \mu_i^*[1\,..\,\tau_{S_1,i}]) > \tau_{S_1,i}$. Hence, $\tau_{S',i} \le \tau_{S_1,i}$ and $R_{S',i} \subseteq R_{S_1,i}$. Since $j' \in M_{S',i}$ and $j \notin \mathsf{MP}(S_1, S_2)$, it must be that $j' \in M_{S_1,i}$. Similarly, $j' \in M_{S_2,i}$. Thus, $j = j' \circlearrowleft n \in (f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)) \cup \mathsf{MP}(S_1, S_2)$. ◁

▷ Claim 5.12. $|\mathsf{C}(S')| \ge \frac{\gamma-2}{2\gamma}n$.

Proof. Recall that $|\mathsf{C}(S_1)| \ge \frac{1}{2}n$. If $\mu_{S_1,i} = \mu_{S',i}$ and $i \in \mathsf{C}(S_1)$, then $i \in \mathsf{C}(S')$. The only indices $i \in \mathsf{C}(S_1) \cap \mathsf{N}(S')$ are indices such that $\mu_{S_1,i} \ne \mu_{S',i}$, which means that $\mathsf{MP}(S_1, S_2) \cap ([i\,..\,i+3\ell-1] \circlearrowleft n) \ne \emptyset$. Hence, each $m \in \mathsf{MP}(S_1, S_2)$ will remove at most 3ℓ positions from $\mathsf{C}(S_1)$. Thus, $|\mathsf{C}(S')| \ge \frac{1}{2}n - |\mathsf{MP}(S_1, S_2)|3\ell \ge \frac{1}{2}n - k\frac{n}{\gamma k} = \frac{\gamma-2}{2\gamma}n$. ◁

Due to Claim 5.12, we have $|\mathsf{C}(S')| \ge \frac{\gamma-2}{2\gamma}n$, and therefore $|f_{\mathsf{c}}(S')| > \frac{\gamma k}{3n}\frac{\gamma-2}{2\gamma}n = \frac{(\gamma-2)k}{6}$ by Lemma 5.9. Due to Claim 5.11, $f_{\mathsf{c}}(S') \subseteq (f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)) \cup \mathsf{MP}(S_1, S_2)$, and therefore $|f_{\mathsf{c}}(S')| \le |\,(f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)) \cup \mathsf{MP}(S_1, S_2)| \le |f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)| + |\mathsf{MP}(S_1, S_2)| \le |f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)| + k$. Consequently, since $\gamma \ge 14$, we have $|f_{\mathsf{c}}(S_1) \cap f_{\mathsf{c}}(S_2)| \ge \frac{\gamma-8}{6}k \ge \frac{14-8}{6}k = k$. ◀

6 Sketches for Pseudo-periodic Strings

Let $\mathcal{H}'_{n,k} \subseteq \Sigma^n$ be the family of $(3\gamma k, (\gamma+1)k)$-pseudo-periodic strings in Σ^n. In this section, we develop circular sketches for $\mathcal{H}'_{n,k}$. We start with a few properties of pseudo-periodic strings. Recall that a string $S \in \Sigma^n$ is called (α, β)-*pseudo-periodic* if it has an (α, β)-base $S' \in \Sigma^n$ with $\mathsf{root}(S') \le \frac{n}{\alpha}$ and $\mathsf{Ham}(S, S') \le \beta$. If $\lfloor\alpha\rfloor > 2\beta$, then the (α, β)-base is unique.

▶ **Lemma 6.1.** *If $S \in \Sigma^n$ is an (α, β)-pseudo-periodic string for some parameters $\lfloor \alpha \rfloor > 2\beta$, then S' has a unique (α, β)-base.*

Proof. Suppose that S has two bases S', S''. Alzamel et al. [3] show that if $|X| = |Y| \geq \mathsf{per}(X) + \mathsf{per}(Y)$ and $X \neq Y$, then $\mathsf{Ham}(X, Y) \geq \left\lfloor \frac{2n}{\mathsf{per}(X)+\mathsf{per}(Y)} \right\rfloor$. Setting $X = S'$ and $Y = S''$, we get a contradiction: $\mathsf{Ham}(S', S'') \geq \left\lfloor \frac{2n}{\mathsf{per}(S')+\mathsf{per}(S'')} \right\rfloor \geq \left\lfloor \frac{2n}{\mathsf{root}(S')+\mathsf{root}(S'')} \right\rfloor \geq \left\lfloor \frac{2n}{n/\alpha+n/\alpha} \right\rfloor = \lfloor \alpha \rfloor > 2\beta \geq \mathsf{Ham}(S, S') + \mathsf{Ham}(S, S'') \geq \mathsf{Ham}(S', S'')$. ◀

Moreover, the triangle inequality immediately yields the following observation.

▶ **Observation 6.2.** *Let $S \in \Sigma^n$ be an (α, β)-pseudo-periodic string and let $T \in \Sigma^n$ be a string such that $\mathsf{Ham}(S, T) \leq k$. Then, T is $(\alpha, \beta + k)$-pseudo-periodic, and every (α, β)-base of S is an $(\alpha, \beta + k)$-base of T.*

Combining Lemma 6.1 with Observations 4.1 and 6.2, we obtain the following corollary.

▶ **Corollary 6.3.** *Let $S_1, S_2 \in \mathcal{H}'_{n,k}$ with $(3\gamma k, (\gamma+1)k)$-bases S'_1 and S'_2, respectively. If, for some $m \in \mathbb{Z}$, we have $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$, then $S'_1 = \mathsf{cyc}^m(S'_2)$.*

Proof. By Observation 6.2, S'_1 is a $(3\gamma k, (\gamma+2)k)$-base of $\mathsf{cyc}^m(S_2)$. Moreover, by Observation 4.1, $\mathsf{cyc}^m(S'_2)$ is a $(3\gamma k, (\gamma+1)k)$-base of $\mathsf{cyc}^m(S_2)$, and thus also a $(3\gamma k, (\gamma+2)k)$-base of $\mathsf{cyc}^m(S_2)$. Since $\lfloor 3\gamma k \rfloor > 2(\gamma+2)k$ due to $\gamma \geq 5$, Lemma 6.1 implies that $S'_1 = \mathsf{cyc}^m(S'_2)$. ◀

6.1 A 0-mismatch Circular Sketch

Both the exact and the $(1 \pm \varepsilon)$-approximation sketches of strings in $\mathcal{H}'_{n,k}$ rely on 0-mismatch circular sketches, which we implement using Karp–Rabin fingerprints.

▶ **Fact 6.4** (Karp–Rabin fingerprints [28])**.** *For every positive integer n, there exists a randomized function $\Phi : \Sigma^n \to \{0,1\}^{O(\log n)}$ such that, for every $S_1, S_2 \in \Sigma^n$, the following holds with high probability: if $S_1 \neq S_2$, then $\Phi(S_1) \neq \Phi(S_2)$.*

Proof. The function Φ is based on a fixed prime number $p \geq \max(\sigma, n^2)$ and a uniformly random $x \in [0\,..\,p-1]$. The function Φ maps a string S to $(\sum_{i=1}^{|S|} x^{i-1} \cdot S[i]) \bmod p$. This way, for every two strings $S_1 \neq S_2$ in Σ^n, we have $\Pr[\Phi(S_1) = \Phi(S_2)] \leq \frac{n}{p} \leq \frac{n}{n^2} = n^{-1}$. ◀

▶ **Lemma 6.5.** *There exists a 0-ECS sketch $(\mathtt{sk}_0, \mathtt{dec}_0)$ for Σ^n of size $O(\log n)$ bits with constant decoding time.*

Proof. The construction relies on a Karp–Rabin fingerprint function Φ. The sketch $\mathtt{sk}_0(S)$ for a string $S \in \Sigma^n$ is defined based on the minimum cyclic rotation of S, denoted $\mathsf{minrot}(S)$, and consists of the following components:

- the fingerprint $\Phi(\mathsf{minrot}(S))$ of the minimum cyclic rotation of S,
- the length $\mathsf{root}(S)$ of the primitive root of S,
- the smallest integer $r \geq 0$ such that $S = \mathsf{cyc}^r(\mathsf{minrot}(S))$.

The decoding function $\mathtt{dec}_0$ is given two sketches $\mathtt{sk}_0(S_1) = (\Phi(\mathsf{minrot}(S_1)), \mathsf{root}(S_1), r_1)$, $\mathtt{sk}_0(S_2) = (\Phi(\mathsf{minrot}(S_2)), \mathsf{root}(S_2), r_2)$, and a shift m. If $\Phi(\mathsf{minrot}(S_1)) \neq \Phi(\mathsf{minrot}(S_2))$, then $S_1 \neq \mathsf{cyc}^m(S_2)$, and thus the function returns ∞. Otherwise, $\mathsf{minrot}(S_1) = \mathsf{minrot}(S_2)$ with high probability, and the implementation proceeds assuming that $\mathsf{minrot}(S_1) = T = \mathsf{minrot}(S_2)$ for a string $T \in \Sigma^n$. In particular, this implies $\mathsf{root}(S_1) = \mathsf{root}(T) = \mathsf{root}(S_2)$. Finally, since $S_1 = \mathsf{cyc}^{r_1}(T)$ equals $\mathsf{cyc}^m(S_2) = \mathsf{cyc}^{m+r_2}(T)$ if and only if $\mathsf{root}(T) \mid (m + r_2 - r_1)$, the function returns 0 or ∞ depending on whether $\mathsf{root}(S_1) \mid (m + r_2 - r_1)$ or not. ◀

6.2 A k-ECS Sketch

▶ **Construction 6.6.** The encoding function $\mathtt{circ}_k : \mathcal{H}'_{n,k} \to \{0,1\}^*$ is defined as follows:

1. Let $\mathtt{sk}_0$ be the 0-mismatch sketch of Lemma 6.5.
2. For $S \in \mathcal{H}'_{n,k}$, the encoding $\mathtt{circ}_k(S)$ stores the sketch $\mathtt{sk}_0(S')$ of the $(3\gamma k, (\gamma+1)k)$-base S' of S and the mismatch information $\mathsf{MI}(S, S')$.

▶ **Proposition 6.7.** *There exists a decoding function which, together with the encoding* $\mathtt{circ}_k$ *of Construction 6.6, forms a k-ECS sketch of $\mathcal{H}'_{n,k}$. The size of the sketch is $\tilde{O}(k)$, and the decoding time is $\tilde{O}(k)$ with high probability.*

Proof. The decoding function is given two sketches $\mathtt{circ}_k(S_1) = (\mathtt{sk}_0(S'_1), \mathsf{MI}(S_1, S'_1))$, $\mathtt{circ}_k(S_2) = (\mathtt{sk}_0(S'_2), \mathsf{MI}(S_2, S'_2))$, and a shift m. By Corollary 6.3, if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \le k$, then $S'_1 = \mathsf{cyc}^m(S'_2)$, and this condition is checked by applying $\mathtt{dec}_0(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2), m)$. If the call returns a non-zero result, then ∞ is returned. Otherwise, $S'_1 = \mathsf{cyc}^m(S'_2)$ holds with high probability. The analysis below is conditioned on this event.

First, $\mathsf{MI}(\mathsf{cyc}^m(S_2), \mathsf{cyc}^m(S'_2))$ is retrieved from $\mathsf{MI}(S_2, S'_2)$ by shifting all the the mismatches. Next, the decoding function retrieves $\mathsf{MI}(S_1, \mathsf{cyc}^m(S_2))$ from $\mathsf{MI}(S_1, S'_1)$ and $\mathsf{MI}(\mathsf{cyc}^m(S_2), \mathsf{cyc}^m(S'_2))$ (using Fact 3.2 and assuming that $S'_1 = \mathsf{cyc}^m(S'_2)$) and returns $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = |\mathsf{MI}(S_1, \mathsf{cyc}^m(S_2))|$. ◀

6.3 An (ε, k)-ACS Sketch

For the pseudo-periodic (ε, k)-ACS sketches, we relax the problem statement; we overcome this relaxation in Section 7. In the *relaxed* (ε, k)-ACS sketch, the distances smaller than $\frac{k}{2}$ do not need to be approximated. More precisely, we require the following:

- if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) < \frac{1}{2}k$, then $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) < \frac{1+\varepsilon}{2}k$,
- if $\frac{1}{2}k \le \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \le k$, then $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) \in (1 \pm \varepsilon)\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$,
- otherwise, $\mathtt{dec}(\mathtt{sk}(S_1), \mathtt{sk}(S_2), m) > (1-\varepsilon)k$.

▶ **Construction 6.8.** The encoding function $\mathtt{circ}_{\varepsilon,k} : \mathcal{H}'_{n,k} \to \{0,1\}^*$ is defined as follows:

1. Let $\mathtt{sk}_0$ be the 0-mismatch sketch of Lemma 6.5.
2. Let $A, B \subseteq [n]$ be two subsets with elements sampled independently with rate $p := \sqrt{\frac{\log n}{\varepsilon^2 k}}$.
3. For $S \in \mathcal{H}'_{n,k}$, the encoding $\mathtt{circ}_{\varepsilon,k}(S)$ stores the sketch $\mathtt{sk}_0(S')$ of the $(3\gamma k, (\gamma+1)k)$-base S' of S and the mismatch information $\mathsf{MI}_{A\cup B}(S, S')$.

▶ **Proposition 6.9.** *There exists a decoding function which, together with the encoding* $\mathtt{circ}_{\varepsilon,k}$ *of Construction 6.8, forms a relaxed (ε, k)-ACS sketch of $\mathcal{H}'_{n,k}$. The size of the sketch is $\tilde{O}(\varepsilon^{-1}\sqrt{k})$, and the decoding time is $\tilde{O}(\varepsilon^{-1}\sqrt{k})$ with high probability.*

Proof. The decoding function is given two sketches $\mathtt{circ}_{\varepsilon,k}(S_1) = (\mathtt{sk}_0(S'_1), \mathsf{MI}_{A\cup B}(S_1, S'_1))$ and $\mathtt{circ}_{\varepsilon,k}(S_2) = (\mathtt{sk}_0(S'_2), \mathsf{MI}_{A\cup B}(S_2, S'_2))$, and a shift m. According to Corollary 6.3, if $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \le k$, then $S'_1 = \mathsf{cyc}^m(S'_2)$, and this condition is checked by applying $\mathtt{dec}_0(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2), m)$. If the call returns a non-zero result, then ∞ is returned. Otherwise, $S'_1 = \mathsf{cyc}^m(S'_2)$ holds with high probability. The analysis below is conditioned on this event.

First, $\mathsf{MI}_{A\cap\mathsf{rot}_n^m(B)}(\mathsf{cyc}^m(S_2), \mathsf{cyc}^m(S'_2))$ is retrieved by filtering and shifting $\mathsf{MI}_{A\cup B}(S_2, S'_2)$. Secondly, $\mathsf{MI}_{A\cap\mathsf{rot}_n^m(B)}(S_1, S'_1)$ is retrieved by filtering $\mathsf{MI}_{A\cup B}(S_1, S'_1)$. Then, the algorithm retrieves $\mathsf{MI}_{A\cap\mathsf{rot}_n^m(B)}(S_1, \mathsf{cyc}^m(S_2))$ combining $\mathsf{MI}_{A\cap\mathsf{rot}_n^m(B)}(S_1, S'_1)$ and $\mathsf{MI}_{A\cap\mathsf{rot}_n^m(B)}(\mathsf{cyc}^m(S_2), \mathsf{cyc}^m(S'_2))$ (using Fact 3.2 and assuming that $S'_1 = \mathsf{cyc}^m(S'_2)$). Since $A \cap \mathsf{rot}_n^m(B)$ is a random subset of $[n]$ with elements sampled independently with rate $\frac{\log n}{\varepsilon^2 k}$, the quantity $\frac{\varepsilon^2 k}{\log n}\mathsf{Ham}_{A\cap\mathsf{rot}_n^m(B)}(S_1, \mathsf{cyc}^m(S_2))$ is a $(1 \pm \varepsilon)$-approximation of $\mathsf{Ham}(S_1, S_2)$ with high probability provided that $\mathsf{Ham}(S_1, S_2) = \Omega(k)$; see Lemma 3.1. ◀

7 Proofs of Main Theorems

In this section, we complete our construction of circular k-mismatch sketches for Σ^n.

▶ **Theorem 1.3.** *There exists a k-ECS sketch for Σ^n of size $\tilde{O}(k)$.*

Proof. Our construction combines the k-ECS sketches of Propositions 6.7 and 4.8. For each string $S \in \Sigma$, if $S \in \mathcal{H}'_{n,k}$, then the sketch contains the sketch of Proposition 6.7, and if $S \in \Sigma^n \setminus \mathcal{H}_{n,k}$, then the sketch contains the sketch of Proposition 4.8. Notice that the sketch contains both components if $S \in \mathcal{H}'_{n,k} \setminus \mathcal{H}_{n,k}$.

For two strings $S_1, S_2 \in \Sigma^n$, given the sketches of S_1 and S_2, the decoder works as follows. If the two sketches contain compatible components (of Proposition 4.8 or of Proposition 6.7), then the decoder uses the decoder corresponding to these components. Otherwise, without loss of generality, it must be that $S_1 \in \mathcal{H}_{n,k}$ and $S_2 \notin \mathcal{H}'_{n,k}$. Thus, by Observation 6.2, $\mathsf{Ham}(S_1, S_2) > k$, and therefore the decoder outputs ∞. The decoding time is $\tilde{O}(k)$. ◀

Similarly, combining the results of Sections 4 and 6 gives $(1+\varepsilon)$-approximate sketches. The proof of the following result mimics the proof of Theorem 1.3 and is given in Appendix B for completeness.

▶ **Proposition 7.1.** *There exists a relaxed (ε, k)-ACS sketch for Σ^n of size $\tilde{O}(\varepsilon^{-2}\sqrt{k})$. Its decoding time is $\tilde{O}(\varepsilon^{-1}\sqrt{k} + \varepsilon^{-2})$ with high probability.*

A simple alternative approach yields smaller sketches when k is large compared to n.

▶ **Construction 7.2.** The encoding function $\mathtt{circ}_{\varepsilon,k} : \Sigma^n \to \{0,1\}^*$ is defined as follows:

1. Let $A, B \subseteq [n]$ be two subsets with elements sampled independently with rate $p := \sqrt{\frac{\log n}{\varepsilon^2 k}}$.
2. For $S \in \Sigma^n$, the encoding $\mathtt{circ}_{\varepsilon,k}(S)$ consists of pairs $(i, S[i])$ for $i \in A \cup B$.

▶ **Proposition 7.3.** *There exists a decoding function which, together with the encoding $\mathtt{circ}_{\varepsilon,k}$ of Construction 7.2, forms a relaxed (ε, k)-ACS sketch of Σ^n. The size of the sketch is $\tilde{O}(\frac{n}{\varepsilon\sqrt{k}})$, and the decoding time is $\tilde{O}(\frac{n}{\varepsilon\sqrt{k}})$ with high probability.*

Proof. The decoder, given the sketches of $S_1, S_2 \in \Sigma^n$ and a shift m, uses Lemma 3.1 to estimate $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ based on $\mathsf{Ham}_{A \cap \mathsf{rot}_n^m(B)}(S_1, \mathsf{cyc}^m(S_2))$. For each $i \in A \cap \mathsf{rot}_n^m(B)$, the decoder retrieves $S_1[i]$ from the sketch of S_1 and $\mathsf{cyc}^m(S_2)[i] = S_2[(i-m) \circlearrowleft n]$ from the sketch of S_2. Since $A \cap \mathsf{rot}_n^m(B)$ is a random subset of $[n]$ with elements sampled independently with rate $\frac{\log n}{\varepsilon^2 k}$, the quantity $\frac{\varepsilon^2 k}{\log n}\mathsf{Ham}_{A \cap \mathsf{rot}_n^m(B)}(S_1, \mathsf{cyc}^m(S_2))$ is a $(1 \pm \varepsilon)$-approximation of $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ with high probability provided that $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = \Omega(k)$; see Lemma 3.1. ◀

▶ **Theorem 1.4.** *There exists an (ε, k)-ACS sketch for Σ^n of size $\tilde{O}(\min(\varepsilon^{-2}\sqrt{k}, \varepsilon^{-1.5}\sqrt{n}))$.*

Proof. An (ε, k)-ACS sketch is obtained by combining $O(\log k)$ relaxed (ε, k')-ACS sketches, where k' ranges over powers of two between 1 and $2k$. Depending on whether $k' \le \varepsilon n$ or not, Proposition 7.1 or Proposition 7.3 is used to implement k'-mismatch sketches. ◀

▶ Remark 7.4. Applying Proposition 7.3 instead of Proposition 7.1 improves the sketch size (for $k \ge \varepsilon n$) but degrades the decoding time. We get two alternatives: $\tilde{O}(\varepsilon^{-2}\sqrt{k})$-size sketches with decoding time $\tilde{O}(\varepsilon^{-1}\sqrt{k} + \varepsilon^{-2})$, and $\tilde{O}(\varepsilon^{-1.5}\sqrt{n})$-size sketches with decoding time $\tilde{O}(\varepsilon^{-1.5}\sqrt{n})$.

References

1. Karl R. Abrahamson. Generalized string matching. *SIAM Journal on Computing*, 16(6):1039–1051, 1987. doi:10.1137/0216067.
2. Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. *Journal of Computer and System Sciences*, 58(1):137–147, 1999. doi:10.1006/jcss.1997.1545.
3. Mai Alzamel, Maxime Crochemore, Costas S. Iliopoulos, Tomasz Kociumaka, Ritu Kundu, Jakub Radoszewski, Wojciech Rytter, and Tomasz Waleń. How much different are two words with different shortest periods. In Lazaros S. Iliadis, Ilias Maglogiannis, and Vassilis P. Plagianakos, editors, *14th International Conference on Artificial Intelligence Applications and Innovations, AIAI 2018, Workshops*, volume 520 of *IFIP Advances in Information and Communication Technology*, pages 168–178. Springer, 2018. doi:10.1007/978-3-319-92016-0_16.
4. Amihood Amir, Moshe Lewenstein, and Ely Porat. Faster algorithms for string matching with k mismatches. *Journal of Algorithms*, 50(2):257–275, 2004. doi:10.1016/S0196-6774(03)00097-X.
5. Alexandr Andoni, Assaf Goldberger, Andrew McGregor, and Ely Porat. Homomorphic fingerprints under misalignments: sketching edit and shift distances. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2013*, pages 931–940. ACM, 2013. doi:10.1145/2488608.2488726.
6. Alexandr Andoni, Piotr Indyk, and Robert Krauthgamer. Earth mover distance over high-dimensional spaces. In Shang-Hua Teng, editor, *19th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2008*, pages 343–352. SIAM, 2008. URL: http://dl.acm.org/citation.cfm?id=1347082.1347120.
7. Alexandr Andoni, Robert Krauthgamer, and Krzysztof Onak. Polylogarithmic approximation for edit distance and the asymmetric query complexity. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010*, pages 377–386. IEEE Computer Society, 2010. doi:10.1109/FOCS.2010.43.
8. Ziv Bar-Yossef, T. S. Jayram, Robert Krauthgamer, and Ravi Kumar. The sketching complexity of pattern matching. In Klaus Jansen, Sanjeev Khanna, José D. P. Rolim, and Dana Ron, editors, *8th International Workshop on Randomization and Computation, RANDOM 2004*, volume 3122 of *LNCS*, pages 261–272. Springer, 2004. doi:10.1007/978-3-540-27821-4_24.
9. Djamal Belazzougui and Qin Zhang. Edit distance: Sketching, streaming, and document exchange. In Irit Dinur, editor, *57th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2016*, pages 51–60. IEEE Computer Society, 2016. doi:10.1109/FOCS.2016.15.
10. Karl Bringmann, Marvin Künnemann, and Philip Wellnitz. Few matches or almost periodicity: Faster pattern matching with mismatches in compressed texts. In Timothy M. Chan, editor, *30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1126–1145. SIAM, 2019. doi:10.1137/1.9781611975482.69.
11. Diptarka Chakraborty, Elazar Goldenberg, and Michal Koucký. Streaming algorithms for embedding and computing edit distance in the low distance regime. In Daniel Wichs and Yishay Mansour, editors, *48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016*, pages 712–725. ACM, 2016. doi:10.1145/2897518.2897577.
12. Timothy M. Chan, Shay Golan, Tomasz Kociumaka, Tsvi Kopelowitz, and Ely Porat. Approximating text-to-pattern Hamming distances. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020*, pages 643–656. ACM, 2020. doi:10.1145/3357713.3384266.
13. Panagiotis Charalampopoulos, Tomasz Kociumaka, and Philip Wellnitz. Faster approximate pattern matching: A unified approach, 2020. arXiv:2004.08350.
14. Raphaël Clifford, Allyx Fontaine, Ely Porat, Benjamin Sach, and Tatiana Starikovskaya. The k-mismatch problem revisited. In Robert Krauthgamer, editor, *27th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016*, pages 2039–2052. SIAM, 2016. doi:10.1137/1.9781611974331.ch142.

15 Raphaël Clifford, Tomasz Kociumaka, and Ely Porat. The streaming k-mismatch problem. In Timothy M. Chan, editor, *30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, pages 1106–1125. SIAM, 2019. doi:10.1137/1.9781611975482.68.

16 Raphaël Clifford and Tatiana Starikovskaya. Approximate Hamming distance in a stream. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016*, volume 55 of *LIPIcs*, pages 20:1–20:14. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.20.

17 Graham Cormode. Data sketching. *Communications of the ACM*, 60(9):48–55, 2017. doi:10.1145/3080008.

18 Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2011. doi:10.1561/1900000004.

19 Michael S. Crouch and Andrew McGregor. Periodicity and cyclic shifts via linear sketches. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *14th International Workshop on Approximation Algorithms for Combinatorial Optimization, APPROX 2011*, volume 6845 of *LNCS*, pages 158–170. Springer, 2011. doi:10.1007/978-3-642-22935-0_14.

20 Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In *Natural Computing Series*, pages 1–87. Springer International Publishing, 2020. doi:10.1007/978-3-030-29414-4_1.

21 Michael J. Fischer and Michael S. Paterson. String matching and other products. In Richard M. Karp, editor, *Complexity of Computation*, volume 7 of *SIAM–AMS Proceedings*, pages 113–125. AMS, 1974.

22 Paweł Gawrychowski and Przemysław Uznański. Towards unified approximate pattern matching for Hamming and L_1 distance. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 62:1–62:13. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.62.

23 Shay Golan, Tsvi Kopelowitz, and Ely Porat. Towards optimal approximate streaming pattern matching by matching multiple patterns in multiple streams. In Ioannis Chatzigiannakis, Christos Kaklamanis, Dániel Marx, and Donald Sannella, editors, *45th International Colloquium on Automata, Languages, and Programming, ICALP 2018*, volume 107 of *LIPIcs*, pages 65:1–65:16. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.ICALP.2018.65.

24 Shay Golan, Tsvi Kopelowitz, and Ely Porat. Streaming pattern matching with d wildcards. *Algorithmica*, 81(5):1988–2015, 2019. doi:10.1007/s00453-018-0521-7.

25 Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950. doi:10.1002/j.1538-7305.1950.tb00463.x.

26 Wei Huang, Yaoyun Shi, Shengyu Zhang, and Yufan Zhu. The communication complexity of the Hamming distance problem. *Information Processing Letters*, 99(4):149–153, 2006. doi:10.1016/j.ipl.2006.01.014.

27 Howard J. Karloff. Fast algorithms for approximately counting mismatches. *Information Processing Letters*, 48(2):53–60, 1993. doi:10.1016/0020-0190(93)90177-B.

28 Richard M. Karp and Michael O. Rabin. Efficient randomized pattern-matching algorithms. *IBM Journal of Research and Development*, 31(2):249–260, 1987. doi:10.1147/rd.312.0249.

29 Subhash Khot and Assaf Naor. Nonembeddability theorems via fourier analysis. *Mathematische Annalen*, 334(4):821–852, 2006. doi:10.1007/s00208-005-0745-0.

30 Tsvi Kopelowitz and Ely Porat. Breaking the variance: Approximating the Hamming distance in $1/\varepsilon$ time per alignment. In Venkatesan Guruswami, editor, *56th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2015*, pages 601–613. IEEE Computer Society, 2015. doi:10.1109/FOCS.2015.43.

31 Tsvi Kopelowitz and Ely Porat. A simple algorithm for approximating the text-to-pattern Hamming distance. In Raimund Seidel, editor, *1st Symposium on Simplicity in Algorithms, SOSA 2018*, volume 61 of *OASICS*, pages 10:1–10:5. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2018. doi:10.4230/OASIcs.SOSA.2018.10.

32 S. Rao Kosaraju. Efficient string matching. Manuscript, 1987.

33 Eyal Kushilevitz, Rafail Ostrovsky, and Yuval Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. *SIAM Journal on Computing*, 30(2):457–474, 2000. doi:10.1137/S0097539798347177.

34 Jelani Nelson. Sketching and streaming algorithms for processing massive data. *ACM Crossroads*, 19(1):14–19, 2012. doi:10.1145/2331042.2331049.

35 Rafail Ostrovsky and Yuval Rabani. Low distortion embeddings for edit distance. *Journal of the ACM*, 54(5):23, 2007. doi:10.1145/1284320.1284322.

36 Benny Porat and Ely Porat. Exact and approximate pattern matching in the streaming model. In *50th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2009*, pages 315–323. IEEE Computer Society, 2009. doi:10.1109/FOCS.2009.11.

37 Ely Porat and Ohad Lipsky. Improved sketching of Hamming distance with error correcting. In Bin Ma and Kaizhong Zhang, editors, *18th Annual Symposium on Combinatorial Pattern Matching, CPM 2007*, volume 4580 of *LNCS*, pages 173–182. Springer, 2007. doi:10.1007/978-3-540-73437-6_19.

38 Jakub Radoszewski and Tatiana Starikovskaya. Streaming k-mismatch with error correcting and applications. *Information and Computation*, 271:104513, 2020. doi:10.1016/j.ic.2019.104513.

39 Tatiana Starikovskaya, Michal Svagerka, and Przemysław Uznański. L_p pattern matching in a stream. In Jarosław Byrka and Raghu Meka, editors, *23rd International Workshop on Approximation Algorithms for Combinatorial Optimization, APPROX 2020*, volume 176 of *LIPIcs*, pages 35:1–35:23. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2020. doi:10.4230/LIPIcs.APPROX/RANDOM.2020.35.

40 David P. Woodruff. Optimal space lower bounds for all frequency moments. In J. Ian Munro, editor, *15th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2004*, pages 167–175. SIAM, 2004. URL: http://dl.acm.org/citation.cfm?id=982792.982817.

A Efficient Shift Distance Decoders

In this section, we develop exact and approximate k-mismatch shift distance sketches with efficient decoding procedures. These sketches use the same encoding functions as the corresponding k-mismatch circular sketches, so we only need to develop the decoding procedures.

Our decoding procedures for shift distance heavily rely on their counterparts for decoding the Hamming distance between S_1 and a fixed rotation of S_2. Hence, each of the following four propositions refers to its counterpart in Section 4 or Section 6.

A.1 Shift Distance Sketches for Non-Pseudo-Periodic Strings

▶ **Proposition A.1** (see Proposition 4.5). *There exists a decoding function which, together with the encoding* $\mathtt{circ}_{\varepsilon,k}$ *of Construction 4.4, forms an* (ε, k)*-ASDS sketch of* $\Sigma^n \setminus \mathcal{H}_{n,k}$*. The decoding algorithm costs* $\tilde{O}(\varepsilon^{-2}k)$ *time with high probability.*

Proof. Our decoding procedure iterates over $i \in f(S_1) \cap A$ and $i' \in f(S_2) \cap B$. For each such pair (i, i'), the procedure retrieves the sketches $\mathtt{sk}_\varepsilon(\mathsf{cyc}^i(S_1))$ and $\mathtt{sk}_\varepsilon(\mathsf{cyc}^{i'}(S_2))$ and recovers a $(1+\varepsilon)$-approximation of $\mathsf{Ham}(\mathsf{cyc}^i(S_1), \mathsf{cyc}^{i'}(S_2))$. The algorithm returns the smallest among the values obtained across all the iterations.

Since $\mathsf{Ham}(\mathsf{cyc}^i(S_1), \mathsf{cyc}^{i'}(S_2)) \geq \mathsf{sh}(S_1, S_2)$, the returned value is at least $(1-\varepsilon)\mathsf{sh}(S_1, S_2)$ with high probability (unless the sketches $\mathtt{sk}_\varepsilon$ fail). Moreover, if $\mathsf{sh}(S_1, S_2) \leq k$ with $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = \mathsf{sh}(S_1, S_2)$ for some integer m, then, as in the proof of Proposition 4.5, with high probability, there is a pair of indices $i \in f(S_1) \cap A$ and $i' \in f(S_2) \cap B$ with $i' = (i+m) \circlearrowright n$. Hence, the returned value is at most $(1+\varepsilon)\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = (1+\varepsilon)\mathsf{sh}(S_1, S_2)$ with high probability. ◀

▶ **Proposition A.2** (see Proposition 4.8)**.** *There exists a decoding function which, together with the encoding* $\mathtt{circ}_k$ *of Construction 4.7, forms a k-ESDS sketch of $\Sigma^n \setminus \mathcal{H}_{n,k}$. The decoding algorithm costs $\tilde{O}(k^2)$ time with high probability.*

Proof. The decoding algorithm first computes the sizes $s_m := |f(S_1) \cap \mathsf{rot}_n^m(f(S_2))|$ for all shifts $m \in [n]$. For this, the algorithm iterates over $i \in f(S_1)$ and $i' \in f(S_2)$ incrementing $s_{(i'-i)\circlearrowright n}$. Next, for each shift $m \in [n]$ with $c_m \geq k$, the algorithm uses the decoding function of Proposition 4.8 to retrieve $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$ (or learn that $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) > k$). Finally, the algorithm returns the smallest among the reported values. (If $c_m < k$ for each $m \in [n]$, then the algorithm returns ∞.)

As for correctness, first note that $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \geq \mathsf{sh}(S_1, S_2)$ holds for each $m \in [n]$, so the returned value is at least $\min(k+1, \mathsf{sh}(S_1, S_2))$ with high probability (unless the decoding procedure of Proposition 4.8 fails). Next, suppose that $\mathsf{sh}(S_1, S_2) = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$. As argued in the proof of Proposition 4.8, $s_m = |f(S_1) \cap f(\mathsf{cyc}^m(S_1))| \geq k$ holds with high probability. Consequently, the decoding procedure of Proposition 4.8 was called for S_1, S_2, and m, resulting in $\mathsf{sh}(S_1, S_2)$ with high probability. Hence, the returned value is at most $\mathsf{sh}(S_1, S_2)$ with high probability.

The decoder iterates over $f(S_1) \times f(S_2)$, which is of size $\tilde{O}(k^2)$ with high probability due to Theorem 4.2. Hence, by the pigeonhole principle there are at most $\tilde{O}(\frac{k^2}{k}) = \tilde{O}(k)$ positions $m \in [n]$ such that $c_m \geq k$. For each such position, the decoding time of Proposition 4.8 is $\tilde{O}(k)$. Thus, the total decoding time is $\tilde{O}(k^2)$. ◀

A.2 Shift Distance Sketches for Pseudo-Periodic Strings

▶ **Lemma A.3** (see Lemma 6.5)**.** *There exists a decoding function* $\mathtt{dec}_0^{\mathsf{sh}}$ *which, together with the encoding* $\mathtt{sk}_0$ *of Lemma 6.5, forms an exact 0-ESDS sketch with constant decoding time.*

Proof. The decoding function, given the sketches $\mathtt{sk}_0(S_1) = (\Phi(\mathsf{minrot}(S_1)), \mathsf{root}(S_1), r_1)$ and $\mathtt{sk}_0(S_2) = (\Phi(\mathsf{minrot}(S_2)), \mathsf{root}(S_2), r_2)$, returns 0 or ∞ based on whether $\Phi(\mathsf{minrot}(S_1)) = \Phi(\mathsf{minrot}(S_2))$ or not. ◀

▶ **Proposition A.4** (see Proposition 6.7)**.** *There exists a decoding function which, together with the encoding* $\mathtt{circ}_k$ *of Construction 6.6, forms a k-ESDS sketch of $\mathcal{H}'_{n,k}$. The decoding algorithm costs $\tilde{O}(k^2)$ time with high probability.*

Proof. The decoding algorithm is given the sketches $\mathtt{circ}_k(S_1) = (\mathtt{sk}_0(S'_1), \mathsf{MI}(S_1, S'_1))$ and $\mathtt{circ}_k(S_2) = (\mathtt{sk}_0(S'_2), \mathsf{MI}(S_2, S'_2))$. First, the algorithm applies $\mathtt{dec}_0^{\mathsf{sh}}(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2))$ of Lemma A.3. If this call returns a non-zero result, then ∞ is returned. Otherwise, for each $m \in [n]$, the algorithm constructs the following sets:

$$\mathsf{P}_m := \mathsf{MP}(S_1, S'_1) \cap \mathsf{MP}(\mathsf{cyc}^m(S_2), \mathsf{cyc}^m(S'_2))$$
$$\mathsf{P}'_m := \mathsf{P}_m \setminus \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$$

For this, the algorithm iterates over $(i, a, b) \in \mathsf{MI}(S_1, S'_1)$ and $(i', c, d) \in \mathsf{MP}(S_2, S'_2)$, adding i to $\mathsf{P}_{(i'-i)\circlearrowright n}$ and, provided that $a = c$, also to $\mathsf{P}'_{(i'-i)\circlearrowright n}$.

For each shift m with $\mathsf{P}_m \neq \emptyset$, the algorithm uses $\mathtt{dec}_0(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2), m)$ of Lemma 6.5. If this call returns a non-zero result, then m is ignored. Otherwise, $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = \mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2) - |\mathsf{P}_m| - |\mathsf{P}'_m|$ is computed. Finally, the algorithm returns the minimum of $\mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2)$ and the smallest among the computed values $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$.

Correctness. By Corollary 6.3, $\mathsf{sh}(S_1, S_2) \leq k$ guarantees $\mathsf{sh}(S'_1, S'_2) = 0$, so the algorithm correctly returns ∞ if $\mathtt{dec}_0^{\mathsf{sh}}(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2))$ yields a non-zero result. Moreover, $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$ guarantees $S'_1 = \mathsf{cyc}^m(S'_2)$, so the algorithm correctly ignores $m \in [n]$ if $\mathtt{dec}_0(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2), m)$ yields a non-zero result. In the following, we assume $\mathsf{sh}(S'_1, S'_2) = 0$ with $S'_1 = \mathsf{cyc}^m(S'_2)$ for all the shifts considered. The latter assumption implies $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = \mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2) - |\mathsf{P}_m| - |\mathsf{P}'_m|$ (compare the proof of Fact 3.2). Moreover, $\mathsf{sh}(S_1, S_2) \leq \mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2)$ holds by the triangle inequality, Hence, the returned value is at least $\mathsf{sh}(S_1, S_2)$ with high probability.

On the other hand, if $\mathsf{sh}(S_1, S_2) = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$ for some shift $m \in [n]$, then $S'_1 = \mathsf{cyc}^m(S'_2)$ and $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = \mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2) - |\mathsf{P}_m| - |\mathsf{P}'_m|$. This either yields $\mathsf{sh}(S_1, S_2) = \mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2)$ (in case of $\mathsf{P}_m = \emptyset$, which yields $|\mathsf{P}_m| = |\mathsf{P}'_m| = 0$) or that m was among the shifts considered (otherwise). In both cases, we conclude that the returned value is at most $\mathsf{sh}(S_1, S_2)$ with high probability. ◀

A relaxed (ε, k)-ASDS sketch is defined analogously to a relaxed (ε, k)-ACS sketch:

- if $\mathsf{sh}(S_1, S_2) < \frac{1}{2}k$, then $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) < \frac{1+\varepsilon}{2}k$,
- if $\frac{1}{2}k \leq \mathsf{sh}(S_1, S_2) \leq k$, then $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) \in (1 \pm \varepsilon)\mathsf{sh}(S_1, S_2)$,
- otherwise, $\mathtt{dec}^{\mathsf{sh}}(\mathtt{sk}(S_1), \mathtt{sk}(S_2)) > (1 - \varepsilon)k$.

▶ **Proposition A.5** (see Proposition 6.9). *There exists a decoding function which, together with the encoding $\mathtt{circ}_{\varepsilon,k}$ of Construction 6.8, forms a relaxed (ε, k)-ASDS sketch of $\mathcal{H}'_{n,k}$. The decoding algorithm costs $\tilde{O}(\varepsilon^{-2}k)$ time with high probability.*

Proof. The decoding algorithm is given sketches $\mathtt{circ}_{\varepsilon,k}(S_1) = (\mathtt{sk}_0(S'_1), \mathsf{MI}_{A\cup B}(S_1, S'_1))$ and $\mathtt{circ}_{\varepsilon,k}(S_2) = (\mathtt{sk}_0(S'_2), \mathsf{MI}_{A\cup B}(S_2, S'_2))$. First, the algorithm applies $\mathtt{dec}_0^{\mathsf{sh}}(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2))$ of Lemma A.3. If this call returns a non-zero result, then ∞ is returned. Otherwise, for each $m \in [n]$, the algorithm constructs the sets $\mathsf{P}_m \cap A \cap \mathsf{rot}_n^m(B)$ and $\mathsf{P}'_m \cap A \cap \mathsf{rot}_n^m(B)$, where

$$\mathsf{P}_m := \mathsf{MP}(S_1, S'_1) \cap \mathsf{MP}(\mathsf{cyc}^m(S_2), \mathsf{cyc}^m(S'_2))$$
$$\mathsf{P}'_m := \mathsf{P}_m \setminus \mathsf{MP}(S_1, \mathsf{cyc}^m(S_2))$$

are defined as in the proof of Proposition A.4. For this, the algorithm iterates over $(i, a, b) \in \mathsf{MI}_A(S_1, S'_1)$ and $(i', c, d) \in \mathsf{MI}_B(S_2, S'_2)$, adding i to $\mathsf{P}_{(i'-i)\circlearrowright n} \cap A \cap \mathsf{rot}_n^{(i'-i)\circlearrowright n}(B)$ and, provided that $a = c$, also to $\mathsf{P}'_{(i'-i)\circlearrowright n} \cap A \cap \mathsf{rot}_n^{(i'-i)\circlearrowright n}(B)$.

For each shift m with $\mathsf{P}_m \cap A \cap \mathsf{rot}_n^m(B) \neq \emptyset$, the algorithm uses $\mathtt{dec}_0(\mathtt{sk}_0(S'_1), \mathtt{sk}_0(S'_2), m)$ of Lemma 6.5. If this call returns a non-zero result, then m is ignored. Otherwise, the algorithm computes

$$d_m := \mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2) - \tfrac{\varepsilon^2 k}{\log n}(|\mathsf{P}_m \cap A \cap \mathsf{rot}_n^m(B)| + |\mathsf{P}'_m \cap A \cap \mathsf{rot}_n^m(B)|).$$

Finally, the algorithm returns the minimum of $\mathsf{Ham}(S_1, S'_1) + \mathsf{Ham}(S_2, S'_2)$ and the smallest among the computed values d_m.

Correctness. By Corollary 6.3, $\mathsf{sh}(S_1, S_2) \leq k$ guarantees $\mathsf{sh}(S_1', S_2') = 0$, so the algorithm correctly returns ∞ if $\mathtt{dec}_0^{\mathsf{sh}}(\mathtt{sk}_0(S_1'), \mathtt{sk}_0(S_2'))$ yields a non-zero result. Moreover, $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$ guarantees $S_1' = \mathsf{cyc}^m(S_2')$, so the algorithm correctly ignores $m \in [n]$ if $\mathtt{dec}_0(\mathtt{sk}_0(S_1'), \mathtt{sk}_0(S_2'), m)$ yields a non-zero result. In the following, we assume $\mathsf{sh}(S_1', S_2') = 0$ with $S_1' = \mathsf{cyc}^m(S_2')$ for all the shifts considered.

Recall that $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) = \mathsf{Ham}(S_1, S_1') + \mathsf{Ham}(S_2, S_2') - |\mathsf{P}_m| - |\mathsf{P}'_m|$ holds provided that $S_1' = \mathsf{cyc}^m(S_2')$. Since $A \cap \mathsf{rot}_n^m(B)$ is a random subset of $[n]$ with elements sampled independently with rate $\frac{\log n}{\varepsilon^2 k}$, the quantity $\frac{\varepsilon^2 k}{\log n}(|\mathsf{P}_m \cap A \cap \mathsf{rot}_n^m(B)| + |\mathsf{P}'_m \cap A \cap \mathsf{rot}_n^m(B)|)$ is with high probability a $\pm\frac{\varepsilon k}{2}$-additive approximation of $|\mathsf{P}_m| + |\mathsf{P}'_m|$ (which can be argued as in the proof of Lemma 3.1). Consequently, the computed value d_m is with high probability a $\pm\frac{\varepsilon k}{2}$-additive approximation of $\mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2))$. As $\mathsf{sh}(S_1, S_2) \leq \mathsf{Ham}(S_1, S_1') + \mathsf{Ham}(S_2, S_2')$ holds by the triangle inequality, this means that the returned value is at least $(1-\varepsilon)\mathsf{sh}(S_1, S_2)$ with high probability provided that $\mathsf{sh}(S_1, S_2) \geq \frac{1}{2}k$.

On the other hand, if $\mathsf{sh}(S_1, S_2) = \mathsf{Ham}(S_1, \mathsf{cyc}^m(S_2)) \leq k$ for some shift $m \in [n]$, then $S_1' = \mathsf{cyc}^m(S_2')$ and d_m is a $\pm\frac{\varepsilon k}{2}$-additive approximation of $\mathsf{sh}(S_2, S_2')$. This either yields $\mathsf{Ham}(S_1, S_1') + \mathsf{Ham}(S_2, S_2') \leq (1+\varepsilon)\mathsf{sh}(S_1, S_2)$ (if $\mathsf{P}_m \cap A \cap \mathsf{rot}_n^m(B) = \emptyset$, which yields $|\mathsf{P}_m \cap A \cap \mathsf{rot}_n^m(B)| = |\mathsf{P}'_m \cap A \cap \mathsf{rot}_n^m(B)| = 0$) or that m was among the shifts considered (otherwise). In both cases, we conclude that the returned value is at most $(1+\varepsilon)\mathsf{sh}(S_1, S_2)$ with high probability. ◀

A.3 Shift Distance Sketches for Σ^n

After handling non-pseudo-periodic and pseudo-periodic strings separately, we derive sketches for the whole Σ^n. The following results provide efficient shift distance decoding procedures for the circular k-mismatch sketches described in Section 7.

For the exact case, using Propositions A.2 and A.4, the same construction as in the proof of Theorem 1.3 yields the following corollary.

▶ **Corollary A.6** (see Theorem 1.3). *There exists a k-ESDS sketch of size $\tilde{O}(k)$ with decoding time $\tilde{O}(k^2)$.*

For the approximate case, using Propositions A.1 and A.5, the same construction as in Proposition 7.1 yields a relaxed (ε, k)-ASDS sketch of size $\tilde{O}(\varepsilon^{-2}\sqrt{k})$ with decoding time $\tilde{O}(\varepsilon^{-2}k)$.

▶ **Proposition A.7** (see Proposition 7.1). *There exists a relaxed (ε, k)-ASDS sketch of size $\tilde{O}(\varepsilon^{-2}\sqrt{k})$ and decoding time of $\tilde{O}(\varepsilon^{-2}k)$.*

The following provides an alternative method for constructing (ε, k)-ASDS sketches which improves the sketch size (for $k \geq \varepsilon n$) but degrades the decoding time.

▶ **Proposition A.8** (see Proposition 7.3). *There exists a decoding function which, together with the encoding $\mathtt{circ}_{\varepsilon,k}$ of Construction 7.2, forms a relaxed (ε, k)-ASDS sketch of Σ^n. The decoding time is $\tilde{O}(\frac{n^2}{\varepsilon^2 k})$ with high probability.*

Proof. The decoding function, given the sketches of $S_1, S_2 \in \Sigma^n$, computes a value $d_m := \mathsf{Ham}_{A \cap \mathsf{rot}_n^m(B)}(S_1, \mathsf{cyc}^m(S_2))$ for each $m \in [n]$. For this, the algorithm iterates over $(i, S_1[i])$ with $i \in A$ (retrieved from the sketch of S_1) and $(i', S_2[i'])$ with $i' \in B$ (retrieved from the sketch of S_2), and increments $d_{(i'-i)\circlearrowright n}$ if $S_1[i] \neq S_2[i']$.

As in the proof of Proposition 7.3, $\frac{\varepsilon^2 k}{\log n}\mathsf{Ham}_{A\cap\mathsf{rot}_n^m(B)}(S_1,\mathsf{cyc}^m(S_2))$ is a $(1\pm\varepsilon)$-approximation of $\mathsf{Ham}(S_1,\mathsf{cyc}^m(S_2))$ with high probability provided that $\mathsf{Ham}(S_1,\mathsf{cyc}^m(S_2))=\Omega(k)$ (and $\frac{\varepsilon^2 k}{\log n}\mathsf{Ham}_{A\cap\mathsf{rot}_n^m(B)}(S_1,\mathsf{cyc}^m(S_2))=o(k)$ otherwise). Hence, the algorithm returns as an approximation of $\mathsf{sh}(S_1,S_2)$ the smallest value $\frac{\varepsilon^2 k}{\log n}\mathsf{Ham}_{A\cap\mathsf{rot}_n^m(B)}(S_1,\mathsf{cyc}^m(S_2))$ among $m\in[n]$. ◀

▶ **Corollary A.9** (see Theorem 1.4). *There exists an (ε,k)-ASDS sketch of size $\tilde{O}(\varepsilon^{-2}\sqrt{k})$ with decoding time $\tilde{O}(\varepsilon^{-2}k)$, and an (ε,k)-ASDS sketch of size $\tilde{O}(\varepsilon^{-1.5}\sqrt{n})$ with decoding time $\tilde{O}(\varepsilon^{-3}n)$.*

B Missing Proofs

▶ **Lemma 3.1.** *Let A be a random subset of $[n]$ with elements chosen independently at rate p. For $0<\varepsilon<1$, we have $\Pr[\mathsf{Ham}_A(S,T)\in(1\pm\varepsilon)p\mathsf{Ham}(S,T)]\geq 1-2\exp\left(-\frac{p\mathsf{Ham}(S,T)\varepsilon^2}{3}\right)$.*

Proof. For each index $i\in[n]$, let x_i be an indicator variable such that $x_i=1$ if $i\in\mathsf{MI}_A(S,T)$ and $x_i=0$ otherwise. Note that $\mathsf{Ham}_A(S,T)=|\mathsf{MI}_A(S,T)|=\sum_{i=1}^n x_i$ and that x_i are independent variables. For every $i\in\mathsf{MI}(S,T)$, we have $\Pr[x_i=1]=p$ and, for every $i\notin\mathsf{MI}(S,T)$, we have $\Pr[x_i=1]=0$. Thus, $\mathbb{E}[\sum_{i=1}^n x_i]=\mathbb{E}[\sum_{i\in\mathsf{MI}_A(S,T)}x_i]=p\mathsf{Ham}_A(S,T)$. Hence, by the Chernoff bound (see, e.g., [20])

$$\Pr\left[\left|\sum_{i=1}^n x_i - p\mathsf{Ham}_A(S,T)\right|>\varepsilon p\mathsf{Ham}_A(S,T)\right]\leq 2\exp\left(-\frac{p\mathsf{Ham}_A(S,T)\varepsilon^2}{3}\right).$$

Thus, $\Pr[\mathsf{Ham}_A(S,T)\in(1\pm\varepsilon)p\mathsf{Ham}(S,T)]\geq 1-2\exp(-\frac{p\mathsf{Ham}(S,T)\varepsilon^2}{3})$. ◀

▶ **Fact 3.2.** *For every $S,T,U\in\Sigma^n$ and every $A\subseteq[n]$, the mismatch information $\mathsf{MI}_A(S,U)$ can be retrieved from $\mathsf{MI}_A(S,T)$ and $\mathsf{MI}_A(T,U)$ in time $\tilde{O}(\mathsf{Ham}_A(S,T)+\mathsf{Ham}_A(T,U))$.*

Proof. For each $i\in A$, we have one of the following four cases:
- if $i\notin\mathsf{MP}(S,T)$ and $i\notin\mathsf{MP}(T,U)$, then $S[i]=T[i]=U[i]$, so $i\notin\mathsf{MP}(S,U)$,
- if $(i,a,b)\in\mathsf{MI}(S,T)$ and $i\notin\mathsf{MP}(T,U)$, then $S[i]=a\neq b=T[i]=U[i]$, so $(i,a,b)\in\mathsf{MI}(S,U)$,
- if $i\notin\mathsf{MP}(S,T)$ and $(i,b,c)\in\mathsf{MP}(T,U)$, then $S[i]=T[i]=b\neq c=U[i]$, so $(i,b,c)\in\mathsf{MI}(S,U)$,
- if $(i,a,b)\in\mathsf{MI}(S,T)$ and $(i,b,c)\in\mathsf{MP}(T,U)$, then $S[i]=a\neq b=T[i]=b\neq c=U[i]$, so $(i,a,c)\in\mathsf{MI}(S,U)$ (if $a\neq c$) or $i\notin\mathsf{MP}(S,U)$ (if $a=c$). ◀

▶ **Theorem 4.3** ($(1\pm\varepsilon)$-approximate sketches, folklore). *There exists a $(1\pm\varepsilon)$-approximate sketch $\mathtt{sk}_\varepsilon$ such that, given $\mathtt{sk}_\varepsilon(S_1)$ and $\mathtt{sk}_\varepsilon(S_2)$ for two strings $S_1,S_2\in\Sigma^n$, one can decode $\mathsf{Ham}(S_1,S_2)$ with a $(1\pm\varepsilon)$-multiplicative error. The sketches use $\tilde{O}(\varepsilon^{-2})$ space, the decoding algorithm is correct with high probability and costs $\tilde{O}(\varepsilon^{-2})$ time.*

Proof. Consider $\mu:\Sigma\to\{0,1\}^\sigma$ defined as $\mu(c)=0^{c-1}10^{\sigma-c}$. For every words u,v, we have $\mathsf{Ham}(\mu(u),\mu(v))=2\cdot\mathsf{Ham}(u,v)$. We then use AMS sketches [2] on $\mu(u)$ and $\mu(v)$ which allow for decoding of ℓ_2 distance $\|\mu(u)-\mu(v)\|_2$. This is enough since, for binary words, the ℓ_2^2 distance coincides with the Hamming distance. We then note that the AMS sketches of $\mu(u)$ and $\mu(v)$ can be computed without explicitly constructing $\mu(u)$ or $\mu(v)$. ◀

▶ **Proposition 7.1.** *There exists a relaxed (ε,k)-ACS sketch for Σ^n of size $\tilde{O}(\varepsilon^{-2}\sqrt{k})$. Its decoding time is $\tilde{O}(\varepsilon^{-1}\sqrt{k}+\varepsilon^{-2})$ with high probability.*

Proof. Our construction combines the (ε, k)-ACS sketch of Proposition 4.5 and the relaxed (ε, k)-ACS sketch of Proposition 6.9. For each strings S, if $S \in \mathcal{H}'_{n,k}$, then the sketch contains the sketch of S by Proposition 6.9 and, if $S \in \Sigma^n \setminus \mathcal{H}_{n,k}$, then the sketch contains the sketch of S by Proposition 4.5. Notice that, for $S \in \mathcal{H}'_{n,k} \setminus \mathcal{H}_{n,k}$ the sketch contains both components.

For any two strings $S_1, S_2 \in \Sigma^n$, given the sketches of S_1 and S_2, the decoder works as follows. If the two sketches contains compatible components (of Proposition 4.5 or of Proposition 6.9), then the decoder uses the decoder corresponding to these components. Otherwise, without loss of generality, it must be that $S_1 \in \mathcal{H}_{n,k}$ and $S_2 \notin \mathcal{H}'_{n,k}$. Thus, by Observation 6.2, $\mathsf{Ham}(S_1, S_2) > k$, and therefore the decoder outputs ∞. ◀

On Guillotine Separability of Squares and Rectangles

Arindam Khan
Indian Institute of Science, Bangalore, India
arindamkhan@iisc.ac.in

Madhusudhan Reddy Pittu[1]
Indian Institute of Technology, Kharagpur, India
pittumadhusudhan@iitkgp.ac.in

Abstract

Guillotine separability of rectangles has recently gained prominence in combinatorial optimization, computational geometry, and combinatorics. Consider a given large stock unit (say glass or wood) and we need to cut out a set of required rectangles from it. Many cutting technologies allow only end-to-end cuts called *guillotine cuts*. Guillotine cuts occur in *stages*. Each stage consists of either only vertical cuts or only horizontal cuts. In k-stage packing, the number of cuts to obtain each rectangle from the initial packing is at most k (plus an additional trimming step to separate the rectangle itself from a waste area). Pach and Tardos [20] studied the following question: Given a set of n axis-parallel rectangles (in the weighted case, each rectangle has an associated weight), cut out as many rectangles (resp. weight) as possible using a sequence of guillotine cuts. They provide a guillotine cutting sequence that recovers $1/(2\log n)$-fraction of rectangles (resp. weights). Abed et al. [1] claimed that a guillotine cutting sequence can recover a constant fraction for axis-parallel squares. They also conjectured that for any set of rectangles, there exists a sequence of axis-parallel guillotine cuts that recovers a constant fraction of rectangles. This conjecture, if true, would yield a combinatorial $O(1)$-approximation for *Maximum Independent Set of Rectangles (MISR)*, a long-standing open problem. We show the conjecture is not true, if we only allow $o(\log\log n)$ stages (resp. $o(\log n/\log\log n)$-stages for the weighted case). On the positive side, we show a simple $O(n\log n)$-time 2-stage cut sequence that recovers $1/(1+\log n)$-fraction of rectangles. We improve the extraction of squares by showing that 1/40-fraction (resp. 1/160 in the weighted case) of squares can be recovered using guillotine cuts. We also show $O(1)$-fraction of rectangles, even in the weighted case, can be recovered for many special cases of rectangles, e.g. fat (bounded width/height), δ-large (large in one of the dimensions), etc. We show that this implies $O(1)$-factor approximation for Maximum Weighted Independent Set of Rectangles, the weighted version of MISR, for these classes of rectangles.

2012 ACM Subject Classification Theory of computation → Packing and covering problems

Keywords and phrases Guillotine cuts, Rectangles, Squares, Packing, k-stage packing

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.47

Category APPROX

Related Version A full version of the paper is available at
`https://www.csa.iisc.ac.in/~arindamkhan/papers/guillotinefull.pdf`.

Acknowledgements We thank Arnab Maiti and Amatya Sharma for helpful discussions.

[1] A part of this work was done when the author was a Narendra undergraduate summer intern at IISc.


© Arindam Khan and Madhusudhan Reddy Pittu;
licensed under Creative Commons License CC-BY



1 Introduction

Cutting stock problem is a well-studied problem in combinatorial optimization, starting from the seminal work of Gilmore and Gomory [13]. Specially, the 2-D variant has received a lot of attention due to its application in practice [14, 27]. In these problems, we need to cut out some required geometric objects under some constraints, from a large source material such as glass, rubber, metal, wood or cloth. One special constraint, guillotine cut (end-to-end cut) emerges naturally from the design of cutting machines. Starting from the initial source material (*piece*), in each step such a cutting sequence takes one of the available pieces and finds an end-to-end cut along a straight line to divide it into two smaller pieces. Ultimately, each of the required objects corresponds to one of the final pieces. These cuts are simple to program due to column generation techniques [10, 26]. Due to lower cost and simple usability, we find many real-world applications of guillotine cuts, such as in crepe-rubber mill [23], glass industry [22], and paper cutting [19].

The problem has also been studied extensively from theoretical viewpoint. Urrutia [25] asked the following question: Given a family of pairwise disjoint compact convex sets on a sheet of glass, is it true that one can always separate out a constant fraction of them using a guillotine cutting sequence? Pach and Tardos [20] answered the question in negative for line segments. They provided a family of n straight line segments where at most $O(n^{\log_3 2})$ line segments can be separated using guillotine cuts. However, they showed that one can always cut out a constant fraction of the input objects, if all objects have roughly the same size (contains a circle of radius r_1 and contained within a circle of radius r_2, for $r_2 \geq r_1 > 0$). For any family of n pairwise disjoint compact convex sets (resp. line segments) in the plane, one can always separate $\Omega(n^{1/3})$ (resp. $\Omega(n^{1/2})$) of members using guillotine cuts. For any family of n pairwise disjoint compact convex sets (resp. line segments) in the plane, one can always separate $\Omega(n^{1/3})$ (resp. $\Omega(n^{1/2})$) of members.

In this paper, we focus on guillotine separability of rectangles and squares. Given a set of n pairwise disjoint axis-parallel open rectangles $\mathcal{R} := \{R_1, R_2, \ldots, R_n\}$ embedded on a square $[0, N] \times [0, N]$, our goal is to separate as many rectangles as possible by a sequence of axis-parallel guillotine cuts. Pach and Tardos [20] showed that $\Omega(n/\log n)$ rectangles can be separated using guillotine cuts. Abed et al. [1] studied the problem for squares. They claimed a guillotine cutting sequence that recovers 1/81-fraction of any set of axis-parallel squares and made the following conjecture:

▶ **Conjecture 1** ([1]). *For any set of n non-overlapping axis-parallel rectangles there is a guillotine cutting sequence with only axis-parallel cuts separating $\Omega(n)$ of them.*

Furthermore, they extend the problem to the weighted case in which each rectangle R_i has an associated weight $p_i \in \mathbb{Q}$ and the goal is to separate a subset of rectangles using guillotine cuts such that the total profit of separated rectangles is maximized. For this weighted version for squares, they claimed recovery of 4/729-fraction of squares, and a $1/2^{O(d)}$-fraction in the weighted case in d-dimensions (where objects are hypercubes). They also showed that a proof of Conjecture 1 will imply an $O(1)$-approximation for maximum independent set of rectangles (MISR), a notoriously difficult NP-hard problem [2, 8]. In maximum weighted independent set of rectangles (MWISR), we are given a set of possibly overlapping axis-parallel rectangles (with associated profit) and the goal to compute a non-overlapping subset of maximum profit. MISR is the cardinality variant, i.e., when all rectangles have equal profit. This connection between MISR and guillotine separability has made the guillotine separability to rise into prominence in recent years [18].

Guillotine cutting for rectangles can also be viewed as a packing problem where all rectangles are packed in such a way that all of them can be cut out using a guillotine cutting sequence. Gilmore and Gomory [13] initiated systematic study of guillotine packing by *k-stage packing*, where each stage consists of either horizontal or vertical guillotine cuts (but not both). Geometric packing problems are a classical well-studied area in approximation algorithms [9, 15]. In 2-D geometric knapsack problem (2GK) [16], we are given a set of rectangular items (with associated profit) and unit square knapsack, and the goal is to pack a subset of items into the knapsack maximizing the total profit. This problem is strongly NP-hard [17], even for squares with unit profit. The present-best approximation ratio is 1.89 [12]. In 2-D strip packing problem (2SP) [11], we are given a set of rectangular items and fixed-width unbounded-height strip, and the goal is to pack all the items into the strip such that the height of the strip is minimized. Kenyon and Rémila gave an APTAS for the problem [14] using a 3-stage packing. In 2-D bin packing problem (2BP), we are given a set of rectangular items and unit square bins, and the goal is to pack all the items into minimum number of bins. The problem is APX-hard [3] and the present best approximation ratio is 1.405 [4].

All these problems have been studied under k-stage packing [21]. Abed et al. [1] have studied 2GK under guillotine cuts and have given a QPTAS for the cardinality case with quasi-polynomially bounded input. Caprara [6] obtained a 2-stage $T_\infty(\approx 1.691)$-approximation for 2BP, and conjectured that the worst-case asymptotic ratio between the optimal 2-stage 2BP and optimal general 2BP is 3/2. Later, Caprara et al. [7] gave an APTAS for 2-stage 2BP and 2-stage 2SP. Afterwards, Bansal et al. [5] showed an APTAS for guillotine 2BP. Seiden and Woeginger [24] gave an APTAS for guillotine 2SP. Both the APTAS for guillotine 2BP and guillotine 2SP are based on the fact that general guillotine 2BP or guillotine 2SP can be approximated arbitrary well by $O(1)$-stage packing, and such $O(1)$-stage packing can be found efficiently. Bansal et al. [4] conjectured the worst-case asymptotic ratio between the best guillotine 2BP and the best general 2BP is 4/3. This conjecture, if true, along with APTAS for guillotine packing [5], will imply a $(4/3 + \varepsilon)$-approximation for 2BP.

1.1 Our contributions

We obtain improved guillotine separability for many classes of rectangles. We show a simple $O(n \log n)$-time algorithm that recovers $1/(\log n + 1)$-fraction of rectangles. The recursive algorithm of [20] can recover $1/(2 \log n)$-fraction of rectangles, but can also take $\Omega(\log n)$-stages in the worst case, whereas our algorithm takes only 2-stages. We define multi-level lines called poles to partition the rectangles into guillotine separable classes and used this technique to recover a constant fraction of rectangles for many classes (see Section 5). Using ternary partitions we show a slightly improved bound of $n/\log_3(n+2)$. We then show that unlike other packing problems (such as 2BP or 2SP), in our problem any guillotine packing can not be approximated arbitrary well by $O(1)$-stages. In particular, we show:

▶ **Theorem 2.** *Any guillotine cutting algorithm with a constant number of stages can recover at most $O(\frac{\log\log n}{\log n})$ fraction of total weight. In order to recover a constant fraction of weight we require $\Omega(\frac{\log n}{\log\log n})$ stages.*

▶ **Theorem 3.** *Any guillotine cutting algorithm with a constant number of stages can extract at most $O(\frac{1}{\log\log n})$ fraction of total rectangles. In order to extract a constant fraction of rectangles, we require $\Omega(\log\log n)$ number of stages.*

For the case of squares, we found bugs in [1]. We could fix them by loosing additional multiplicative factor. Then using a more involved sampling and exploiting structural properties of guillotine cuts, we obtain further improvement. These structural properties might find usage in related guillotine packing problems.

▶ **Theorem 4.** *For axis-parallel squares, always there exists a guillotine cutting sequence that recovers* $1/40$ *(resp.* $1/160$ *in the weighted case)-fraction.*

We also show that $O(1)$-fraction of rectangles can be recovered for many special classes of rectangles, such as (a) *Fat*: when for each rectangle the ratio of width and height is in $[1/\beta, \beta]$, for some constant β. Fat objects generalize squares and are well-studied [20], (b) *δ-large*: when each rectangle has either width $\geq \delta N$ or height $\geq \delta N$. These rectangles are well-studied in the context of MISR [2], (c) *part-similar*, (d) *anti-laminar* (see Section 5). We also show that if $1/c$-fraction of weight can be recovered for a class of rectangles, then we obtain an $O(n^5)$-time c-approximation for MWISR. Thus obtaining $O(1)$-approximation for MWISR for the above classes of rectangles.

1.2 Organization of the paper

Section 2 describes some building blocks and known results used in this article. Section 3 gives improved guillotine separability of squares. Section 4 describes algorithms and hardness for rectangles. Section 5 studies several special classes of rectangles and obtains constant factor extraction for them. Due to space constraints, some proofs are deferred to the full version .

2 Preliminaries

Let $\mathbb{Z}^+$ (resp. $\mathbb{Z}^{\geq 0}$) be the set of positive (resp. nonnegative) integers. Let us define $[n] := \{1, 2, \ldots, n\}$ and $[n \cup 0] := \{0, 1, 2, \ldots, n\}$, for $n \in \mathbb{Z}^+$. We are given a set of axis-parallel nonoverlapping rectangles $\mathcal{R} := \{R_1, R_2, \ldots, R_n\}$ embedded in a box $K := [0, N] \times [0, N]$. Each rectangle $R_i \in \mathcal{R}$ has width w_i, height h_i, and bottom-left corner at (x_i, y_i). Thus rectangle R_i is defined by the region $R(i) := [x_i, x_i + w_i] \times [y_i, y_i + h_i]$, where $x_i \geq 0, y_i \geq 0, x_i + w_i \leq N, y_i + h_i \leq N$. For any two input rectangles $R_i, R_j (j \neq i)$, we have $R(i) \cap R(j) = \emptyset$. Rectangle R_i has an associated weight p_i. In the unweighted case, all rectangles have the same (unit) weight.

2.1 Guillotine separability

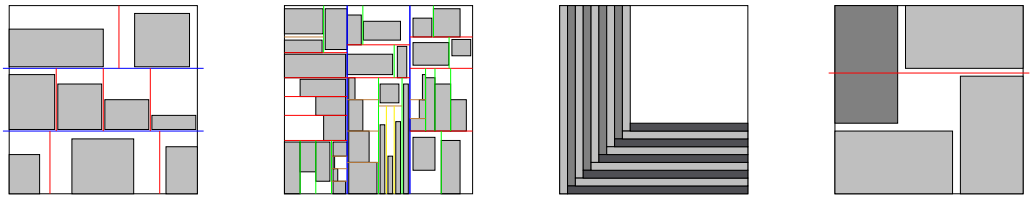
▶ **Definition 5** (Piece)**.** *A* piece *is an axis-aligned rectangular region with axis-parallel rectangles $\mathcal{R}$ embedded on it.*

▶ **Definition 6** (Guillotine cut)**.** *A* guillotine cut *for a piece P is an end-to-end axis-parallel cut along a straight line ℓ dividing the piece into two (rectangular) subpieces P_1 and P_2.*

We define a *stage* of guillotine cuts as a set of end to end, equal, axis-parallel cuts separating piece P into further sub-pieces. Cuts in alternate stages alternate between vertical (parallel to y-axis) and horizontal (parallel to x-axis) cuts.

A guillotine cutting strategy is represented by two types of trees.

▶ **Definition 7** (Guillotine binary tree)**.** A guillotine binary tree *for a set of rectangles $\mathcal{R}$ is a binary tree $\mathcal{T}_B$ where each non-leaf node $v \in V(\mathcal{T}_B)$ is equipped with a piece P_v and the straight line ℓ_v corresponding to the cut such that cutting P_v along the straight line ℓ_v gives us*



(a) 2-stage. **(b)** 5-stage. **(c)** many stage. **(d)** non-guillotine.

Figure 1 Different types of packing: (a), (b), (c) are guillotine packing, while (d) is not.

subpieces P_{v_1} and P_{v_2}, where v_1, v_2 are the children of v. Thus the piece P_r corresponding to the root node r contains $\mathcal{R}$ and for each leaf node v, the piece P_v contains only one rectangle. For each non-leaf node v, let $cutrec(v)$ (resp. $cutwt(v)$) denote the number (resp. total weight) of rectangles cut by line ℓ_v.

▶ **Definition 8** (Guillotine stage tree). *A guillotine stage cutting strategy for a set of rectangles $\mathcal{R}$ is represented by a* guillotine stage tree *$\mathcal{T}_S$ where each non-leaf node $v \in V(\mathcal{T}_S)$ is equipped with a piece P_v and the set of straight lines $\mathcal{L}_v$ corresponding to the stage of the cut such that cutting P_v along the straight lines in $\mathcal{L}_v$ gives us subpieces $P_{v_1}, P_{v_2}, \ldots, P_{v_{|\mathcal{L}_v|}}$, where $v_1, v_2, \ldots, v_{|\mathcal{L}_v|}$ are the children of v. For each non-leaf node $v \in V(\mathcal{T}_S)$, let $cutrec(v)$ (resp. $cutwt(v)$) denote the number (resp. weights) of rectangles cut by lines in $\mathcal{L}_v$. A guillotine cutting strategy has k-stages if the tree $\mathcal{T}_S$ has height k.*

We say that $\mathcal{T}_B$ (resp. $\mathcal{T}_S$) separates a set of rectangles $\mathcal{R}$ if $cutrec(v) = 0$ for all $v \in V(\mathcal{T}_B)$ (resp. $\mathcal{T}_S$).

▶ **Definition 9** (k-good cut). *A cut ℓ_v cutting P_v is a k-good cut, if it intersects at most k rectangles and each side contains at least one rectangle completely.*

▶ **Definition 10** ((k, c)-good cut). *For any piece P having at least $2c$ rectangles, a (k, c)-good cut is a cut which cuts at most k rectangle and also has at least c rectangles on both sides.*

▶ **Definition 11** (Extraction factor). *A guillotine cutting strategy is said to have an extraction factor f if it separates f-fraction of rectangles (resp. weight).*

2.2 Grid sampling for squares

Here we briefly discuss grid sampling from [1] (but modify slightly to fix the bugs) when all input rectangles in $\mathcal{R}$ are squares. Let $len(i)$ be the side length of square R_i. Abed et al. [1] introduced a collection of (multi-level) grid lines to do a four step sampling to show that a constant fraction of squares are guillotine separable. First, we define these gridlines and related notions as they constitute an important component in our proofs too. A square $R_i \in \mathcal{R}$ is said to be at level-i if $len(i) \in \left(\frac{N}{2^{i+1}}, \frac{N}{2^i}\right]$. We independently pick random numbers $\tilde{x}, \tilde{y} \in [0, N)$ to define a random shift for drawing the grid. The vertical grid lines at level-i are drawn at $\tilde{x}, \tilde{x} + \frac{N}{2^i}, \tilde{x} + \frac{2N}{2^i}, \cdots$. Similarly, the horizontal grid lines at level-i are drawn at $\tilde{y}, \tilde{y} + \frac{N}{2^i}, \tilde{y} + \frac{2N}{2^i}, \cdots$ (Note that the coordinates mentioned for the vertical and horizontal grid lines are taken *modulo* N). Grid cells circumscribed by successive grid lines at level-$(i-1)$ are said to be at level-i. Thus the level-i grid cells are square regions of size $\frac{2N}{2^i} \times \frac{2N}{2^i}$. Hence, if a level-$j$ square is completely contained within a level-i grid cell, then $i \leq j + 1$. Note that higher the level, more fine grained the grid is, and smaller the grid cells and squares are.

In the *first* step of sampling, we pick squares from either even levels or odd levels, randomly. Let $\mathcal{R}_0$ be the set of squares remaining afterwards. Then $\mathbb{P}[R \in \mathcal{R}_0] = 1/2$.

In the *second* step of sampling, a square $R \in \mathcal{R}_0$ is removed if it intersects with grid lines in the level below it, i.e., any level-i ($i \geq 1$) square R is removed if it intersects a gridline of level $0, \ldots, i-1$. Let $\mathcal{R}_1$ be the set of squares that remained after this step.

▶ **Lemma 12** ([1]). *A level-i square $R \in \mathcal{R}_0$ of side length $len(R) \in \left(\frac{N}{2^{i+1}}, \frac{N}{2^i}\right]$ remains in $\mathcal{R}_1$ with probability $(1-\mu_R)^2 \geq 1/4$, where $\mu_R = len(R) \cdot \frac{2^{i-1}}{N}$*

In the *third* sampling step, the squares in $\mathcal{R}_1$ are sampled to obtain $\mathcal{R}_2$ so that each level-i grid cell contains at most one square of level-i. Let $\mathcal{R}_1^C$ denote the subset of level-i squares contained within a grid cell C at level-i. We sample each $R \in \mathcal{R}_1^C$ with probability $\{(1-\mu_R)^2 \cdot M_C)^{-1}$ for $M_C = \sum_{S \in \mathcal{R}_1^C} (1-\mu_S)^{-2}$. Let $\mathcal{R}_2$ be the set of squares remaining after this process. Then the probability that a level-i square R remains in $\mathcal{R}_2$ is: $\mathbb{P}[R \in \mathcal{R}_0] \cdot \mathbb{P}[R \in \mathcal{R}_1 | R \in \mathcal{R}_0] \cdot \mathbb{P}[R \in \mathcal{R}_2 | R \in \mathcal{R}_1] = \frac{1}{2} \cdot (1-\mu_R)^2 \cdot \left(\frac{1}{(1-\mu_R)^2 \cdot M_C}\right) = 1/(2 \cdot M_C)$

In the *fourth* step, squares in $\mathcal{R}_2$ are sampled further to obtain a guillotine separable set.

▶ **Definition 13** (ε-guillotine sampling). *An* ε-guillotine sampling *for objects $\mathcal{O}$ is a distribution $\mathcal{D} : 2^{\mathcal{O}} \to [0,1]$ such that any object $r \in \mathcal{O}$ is sampled by $\mathcal{D}$ with probability at least ε and each subset in the support of $\mathcal{D}$ is guillotine separable.*

▶ **Lemma 14** ([1]). *For any set of objects $\mathcal{O}$, the following two statements are equivalent:*

- **(i)** *there is an ε-guillotine sampling for $\mathcal{O}$,*
- **(ii)** *for any $w : \mathcal{O} \to \mathbb{Z}_{\geq 0}$, there is a guillotine seperable subset $\mathcal{O}' \subseteq \mathcal{O}$ with $w(\mathcal{O}') \geq \varepsilon \cdot w(\mathcal{O})$.*

Let $level(R)$ be the level of square R. For a level-i square R, let $cell(R)$ be the level-i gridcell containing R. We say that two squares R and S are *conflicting* if either R overlaps the boundary of $cell(S)$ or S overlaps the boundary of $cell(R)$. Note that if R overlaps the boundary of $cell(S)$, then $level(R) < level(S)$. A conflict graph H encodes the conflict structures between squares, where the vertex set $V(H)$ corresponds to the squares in $\mathcal{R}$, and there is an edge between squares R and S if and only if R and S are conflicting.

▶ **Lemma 15** ([1]). *For an independent set $I \subseteq V(H)$, $\{R\}_{R \in I}$ are guillotine separable.*

Let H be the conflict graph defined by the squares in $\mathcal{R}_2$. If H is χ-colorable, then we obtain a guillotine separable set of size at least $|\mathcal{R}_2|/\chi$. Abed et al. [1] showed that H is 9-colorable and showed $M_C \leq 81/4$. This shows $\frac{1}{9 \cdot (2M_C)} \geq \frac{2}{729}$-fraction of squares are guillotine separable. For the unweighted case, they claimed an improved bound of $1/81$, by exploiting the structure of the tree representing the binary cutting strategy. However, there are bugs in both the proofs of weighted and unweighted cases.

3 Improved guillotine separability of squares

In this subsection, we prove extraction for unweighted and weighted squares to be $\frac{1}{40}$ and $\frac{1}{160}$, respectively. First we prove the following structural property to showing 5-colorability of conflict graph H. We say that a set of squares cover the edges of a level-l grid cell if the level-l cell has atleast one level-l square inside it and every square in the set intersects atleast one of the edges of the grid cell.

▶ **Lemma 16.** *The edges of any level-l grid cell can be covered by at most 4 squares after the second sampling.*

Proof. Other than squares covering the corners, there can not be any squares on the edges because any square on the edge has to be of level $\leq l-1$ because it intersects a level $l-1$ grid line. But since by first sampling we picked either odd or even parity levels the square on the edge has to be of level $\leq l-2$ which implies that the side of this square exceeds the side length of the level-l grid cell. ◀

▶ **Lemma 17.** *The conflict graph H is 5-colorable.*

Proof. Using similar proof as in [1], we prove this by induction on the number of vertices (squares). The base case (when there is only one square) is obvious. When there are at least two squares, consider the smallest square and the squares adjacent to it in the graph H. By Lemma 16, the vertex corresponding to smallest square has degree at most 4. Now assuming the graph without this vertex is 5-colorable, we can add this vertex and color it with one of the available 5-colors because the degree of this vertex is at most four. So inductively it is proven that the graph H is 5-colorable. ◀

From Lemma 15, the squares representing any independent set of graph H are guillotine separable. The first three (modified) steps of sampling selects every square with probability at least $\varepsilon = 2/81$. Now by sampling the five independent sets from H, uniformly at random, we obtain a set of squares which is ε-guillotine samplable for $\varepsilon = \frac{2}{81} \cdot \frac{1}{5} = 2/405$. From Lemma 14, we obtain extraction factor of $2/405$.

3.1 Further improvement in the unweighted case

We can see that the initial set of squares were sampled thrice after picking a parity of levels and drawing a random grid so that no square of level-l was intersected by a grid line of level less than l and every level-l grid cell has at most one square of level-l inside it. We shall do the third sampling a bit differently. After we draw random grid lines and remove all the squares intersected by grid lines of level lower than that of the square, we now allow up to 6 squares inside a grid cell. Let the set of squares that remained after the first sampling be $\mathcal{R}_1$. If the length of a square $R \in \mathcal{R}$ be $len(R) \in (N/2^{i+1}, N/2^i]$, then the probability that this square stays in $\mathcal{R}_1$ is $(1-\mu_R)^2$ where $\mu_R = len(R) \cdot 2^{i-1}/N$. Let $|C|$ denote the number of squares in cell C. Now for the third sampling let us sample each square from a cell C with a probability $\frac{\min(6,|C|)}{(1-\mu_R)^2 \cdot M_C}$, for $M_C = \sum_{S \in C} \frac{1}{(1-\mu_S)^2}$. Let the set of squares that remained after this third sampling be $\mathcal{R}'_2$. Now the probability that a square remains in $\mathcal{R}'_2$ after the third sampling can be written as: $\mathbb{P}[R \in \mathcal{R}_0] \cdot \mathbb{P}[R \in \mathcal{R}_1 | R \in \mathcal{R}_0] \cdot \mathbb{P}[R \in \mathcal{R}_2 | R \in \mathcal{R}_1]$ $= \frac{1}{2} \cdot (1-\mu_R)^2 \cdot \left(\frac{\min(6,|C|)}{(1-\mu_R)^2 \cdot M_C} \right) = \frac{1}{2} \cdot \frac{\min(6,|C|)}{M_C}$.

▷ Claim 18. $M_C / \min(6, |C|) \leq 4$

Proof. Let us find the maximum possible value of $\sum_{r \in C} \frac{1}{(1-\mu_R)^2 \cdot \min(6,|C|)}$ with $\mu_R \in (1/4, 1/2]$.

As $len(R) > len(C)/4$, we have $|C| \leq 9$. As the sum of the areas of squares in C is less than that of the area of C, we have $\sum_{R \in C} {\mu_R}^2 \leq 1$. For the case when $|C| \leq 6$, we have the sum to be at most $\frac{|C|}{(1-1/2)^2 \cdot \min(6,|C|)} = 4$. For $|C| \geq 7$, we use the fact from [1] that the function above is maximized for a given $|C|$ when all μ_R are equal. Thus a simple calculation gives that the maximum occurs when $|C| = 9$ and the value of the summation is equal to $81/24 < 4$. ◁

Therefore, at the end of this sampling, each square is left with probability at least $1/8$. Now we consider further properties of guillotine cuts to obtain a better guillotine separable set.

▶ **Definition 19** (T-cut). *A T-cut is constituted by two axis-parallel line segments ℓ_A and ℓ_B such that one of the end points of ℓ_B lies on ℓ_A and $\ell_A \perp \ell_B$.*

A set of rectangles is said to be intersected by a T-cut if each rectangle in the set has a non-empty intersection with the T-cut. The following observations will be helpful in proving the existence of good cuts.

▶ **Observation 20.** *Any set of rectangles intersected by a T-cut is guillotine separable.*

Proof. W.l.o.g. assume in the T-cut ℓ_B is vertical and the bottom endpoint is lying on horizontal line segment ℓ_A. Consider the cut along the bottom edge of the topmost rectangle that is intersected by ℓ_B. If this line is a guillotine cut then we are done, otherwise consider the line aligned with the edge of the rectangle that got intersected first by this line. This has to be a guillotine cut otherwise will contradict our assumption about the top most rectangle. ◀

▶ **Observation 21.** *If there are at most 3 rectangles in a piece, they are guillotine separable.*

Proof. If there are no axis-parallel line intersecting at least two rectangles, the rectangles are guillotine separable by guillotine cuts along the edges of rectangles. Otherwise, if there exists such a line ℓ, we can extend a perpendicular line segment to ℓ that intersects the third remaining rectangle. This will form a T-cut and the proof follows from Observation 21. ◀

▶ **Observation 22.** *If there are at most n (≤ 6) rectangles in a piece, then we can separate at least $(n-1)$ squares by a sequence of guillotine cuts.*

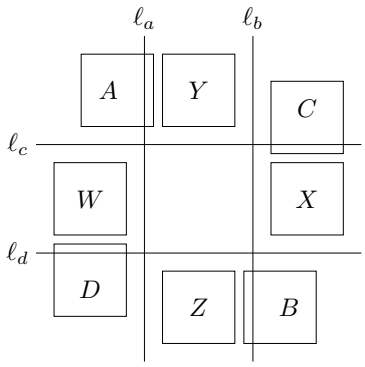
Proof. W.l.o.g. assume that there is no 0-good cut in the piece.
Case 1: There is a 1-good cut ℓ. Let R be the square intersected by ℓ. If there are at most three rectangles on both sides of this cut then we can extract all rectangles, except the one cut by ℓ, due to Observation 21. The other case is when there is one rectangle on one side and four on the other side. If those four rectangles are guillotine separable then we are done. Otherwise, w.l.o.g. assume ℓ is vertical and these four rectangles are on the right side of ℓ and a single rectangles R' is on the left side of ℓ. Let ℓ_1 be the line aligned with the right edge of R'. Then ℓ_1 should intersect R as there are no 0-good cuts. W.l.o.g. assume R lies above R'. Let ℓ_2 be the line aligned with the bottom edge of R and ℓ_3 be the line aligned with the left most edge of the first rectangle (from the left) that ℓ_2 intersects. We can see that ℓ_3 is again a 1-good cut and cut along ℓ_2 after cutting along ℓ_3 is a 0-good cut. These two cuts seperated the piece into three parts with atmost three rectangles in each sub-piece by cutting atmost one rectangle. By Observation 21 we are done.
Case 2: There is no 1-good cut in the piece. Let ℓ_1 be the line aligned with the left most right edge of the rectangles. This line has at least two rectangles on it as there is no 1-good cut. If we have at least four rectangles intersected by ℓ_1 then we are done because we can form a T-cut with all rectangles except possibly one rectangles and by Observation 20 we can extract all of them. Now consider if we have exactly three rectangles on ℓ_1. Then consider the line ℓ_2 that is aligned with the bottom edge of the top most rectangle on ℓ_1. This line should have at least two rectangles on it by our assumption which will again lead to a T containing five rectangles and we are done. Now suppose ℓ_1 has exactly two rectangles on it and ℓ_2 also has exactly two rectangles on it. Let ℓ_3 be the line aligned with the left edge of the first rectangle intersected by ℓ_2. For ℓ_3 to also intersect two rectangles, one of the rectangles have to be the lower rectangle that ℓ_1 intersects. However this implies that the cut along the top edge of the bottom rectangle on ℓ_1 is a 1-good cut which is a contradiction. ◀

▶ **Observation 23.** *If there are at most* 10 *rectangles, then there exists a* $(4,1)$*-good cut.*

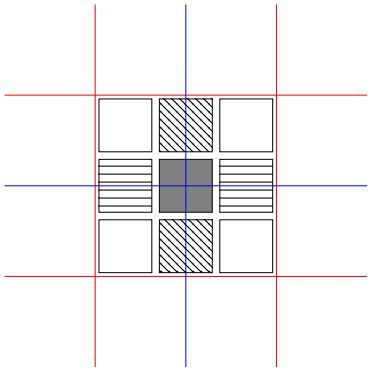
Proof. Consider the median cut that has almost equal number of rectangles on each side and let the line corresponding to the cut be ℓ_1. Suppose if the cut has at least six rectangles on it then any cut separating these rectangles cuts at most four rectangles and hence is a $(4,1)$ good cut. If ℓ_1 cuts at most four then it itself is a $(4,1)$ good cut. The only case is when it has exactly five rectangles on it. Consider every cut separating these five rectangles. If none of these cuts are $(4,1)$-good cuts then every cut has to intersect all other remaining rectangles. This implies that there is one guillotine stage cut separating all the five rectangles on ℓ_1 from the rest. ◀

▶ **Observation 24.** *If there are at most* 12 *squares with level less than* l *in a piece* P *and at least* 7 *squares lie on a grid line of level-*$(l-1)$ *then there exists a* $(6,2)$*-good cut in* P*.*

Proof. W.l.o.g. assume that the level-$(l-1)$ gridline ℓ_h that is intersecting at least 7 squares, is horizontal. Consider the leftmost level-$(l-1)$ vertical line ℓ_v with at least 2 squares on its left. We now prove that ℓ_v is a $(6,2)$-good cut. Note that this vertical line can cut at most 6 squares because there are a total of at most 12 squares. Let ℓ'_v be the vertical line of level-$(l-1)$ that is immediately to the left of ℓ_v. Then ℓ'_v can have at most 1 square to the left of it by definition of line ℓ_v. We know that there can be at most 1 square of level $< l$ in the space between ℓ_v and ℓ'_v. Thus there are at most 3 squares to the left of line ℓ_v. Therefore, we obtain ℓ_v to be a $(6,2)$-good cut. ◀



(a) Different classes of squares after sampling in the unweighted case.



(b) Different third sampling for weighted case (squares with diagonal, horizontal, full, and no shading belong to classes X, Y, Z, W, respectively).

Figure 2

▶ **Lemma 25.** *Using the new sampling scheme, every sampled instance in the piece (and the following subpieces) admits either a* $(4,1)$*-good cut or a* $(6,2)$*-good cut.*

Proof. For contradiction, assume that there are no $(4,1)$ and $(6,2)$-good cuts. Let P' be the considered piece. Now there are two cases:
Case 1: P' *has at least* 13 *squares.* Label the largest 13 squares as $R_1, R_2, \ldots, R_{12}, R_{13}$ in non-increasing order of size. Consider the grid cell C_{13} of square R_{13} (say, of level l). Let the left, right, top and bottom grid lines forming the edges of C_{13} be $\ell_a, \ell_b, \ell_c, \ell_d$, respectively (see Figure 2a). Let us also label the set of squares that are intersected by $\ell_a, \ell_b, \ell_c, \ell_d$ as A, B, C, D, respectively. Also let us label the set of squares lying completely to the left

of line ℓ_a as W, right of line ℓ_b as X, above line ℓ_c as Y, and below line ℓ_d as Z. We observe that at least one of the sets W, X, Y, Z are non empty because there can be at most six squares inside C_{13} after the sampling, and at most four squares are in the set $(A \cup B \cup C \cup D) \setminus (W \cup X \cup Y \cup Z)$ (squares intersecting the edges of the cell). W.l.o.g. let the largest non-empty set be W.
First, we consider the case when $|W| = 1$. As we don't have $(4, 1)$-good cuts, ℓ_a intersects at least 5 squares. So, $|Y \cup Z| \geq 2$. However, $|Y|, |Z| \leq |W|$. Hence, $|Y| = 1$ and $|Z| = 1$. Then both the lines ℓ_c and ℓ_d should intersect 5 squares each. Which implies $|X| \geq 2$ which is a contradiction to our assumption that $|W| = 1$ is the largest among $|W|, |X|, |Y|, |Z|$. Now we consider the case when $|W| \geq 2$. If there is any square on the right of ℓ_a other than R_{13} then consider ℓ_a can not intersect ≤ 6 squares, as then we obtain a $(6, 2)$-good cut. On the other hand if it intersects at least 7 squares, then from Observation 24 we have a $(6, 2)$ good cut which is a contradiction. So let us consider the case when there are no squares on the right of ℓ_a except R_{13}. As there are no $(4, 1)$-good cuts, ℓ_a should intersect at least 5 squares. Since we have at least 5 squares on ℓ_a, $|Y \cup Z| \geq 2$. W.l.o.g let us assume $|Y| \geq 1$. Then it implies again that there have to be at least 5 squares on line ℓ_c and all these squares are on or to the left of line ℓ_a. Using similar arguments as in proof of Observation 24, let ℓ_p be the left most level-$(l-1)$ vertical grid line that has at least two squares on its left. Note that $\ell_p \neq \ell_a$ and thus has at least two squares on its right. Then ℓ_p must intersect ≥ 7 squares as there are no $(6, 2)$-good cuts. However, then by observation 24, we will again have a $(6, 2)$-good cut, giving a contradiction.
Case 2: P' has at most 12 *squares.* Consider any two squares and draw a line ℓ_1 that separates both these squares. The number of squares lying on this line have to be at least 5 by our assumption that there is no $(4, 1)$-good cut. Now draw a line ℓ_2 perpendicular to ℓ_1 which has at least two squares on both its sides out of the 5 squares that were lying on ℓ_1. The line ℓ_2 should have at least 7 squares on it to avoid being the $(6, 2)$-good cut. Now consider the line ℓ_3 perpendicular to ℓ_2 that has at least 2 squares on each side of ℓ_3 out of the 7 squares lying on l_2. Line ℓ_3 also should have at least 7 squares lying on it by the previous argument. Two perpendicular lines each having 7 squares lying on them guarantee a total of at least 13 squares in total. This is a contradiction. ◀

Using these observations and several other properties of guillotine cuts, we show existence of good cuts in the sample instance.

▶ **Theorem 26.** *Given a set of n squares obtained after the sampling, we can find a subset of at least $n/40$ squares that are guillotine separable.*

Proof. Using Lemma 25, we define a guillotine cutting sequence on a piece using only $(4, 1)$ and $(6, 2)$-good cuts until each subpiece has 6 or fewer rectangles. Then if the subpiece has $4, 5$ or 6 rectangles and is not guillotine separable, we use Observation 22 to separate them, cutting at most one rectangle. This whole cutting strategy can be represented by a binary tree with internal nodes storing the number of squares that were killed by the cut. Each leaf node contains guillotine separable squares. Let f_1, f_2 be the number of leaf nodes containing one square and greater than one square, respectively. By the property of binary tree we know that the number of internal nodes is $f_1 + f_2 - 1$. Let v_i be a leaf node and v_j be its parent node. Now if v_i has one square in it and v_j corresponds to a $(4, 1)$-good cut, then v_j has at most 4 squares in it. If v_i was obtained using Observation 22 then its parent node v_j again has only one square in it. Now assume that there is an internal node v_a which has two leaf children v_b, v_c, and both v_b and v_c have exactly one square in each of them. Then as v_a can have at most four squares, the three nodes v_a, v_b, v_c have at most six squares in

total. By our assumption then we would have used the strategy as defined in Observation 22. This gives a contradiction. Thus every internal node can have at most one leaf child which has 1 square in it. This observation tells us that there are at least f_1 internal nodes each of which correspond to at most 4 squares. The maximum number of rectangles killed is at most $4f_1 + 6(f_2 - 1)$ and at least $f_1 + 2f_2$ squares are extracted. So the fraction of squares that are saved is at least $\frac{f_1+2f_2}{5f_1+8f_2-6} \geq 1/5$. This implies an overall extraction factor of $\frac{1}{8} \cdot \frac{1}{5} = \frac{1}{40}$. ◀

3.2 Further improvement in the weighted case

From Claim 18, The probability for a square to survive after the second sampling is at least 1/8. Now we will divide these squares into four groups and select a group uniformly at random. Every level-l grid cell contains at most six squares of level-l due to the property of second sampling. Now, consider a particular cell C, then assume ℓ_v, ℓ_h be the vertical line and the horizontal line, respectively, that bisect the cell (see Figure 2b). Note that ℓ_v (resp. ℓ_h) belongs to the vertical (resp. horizontal) gridlines of level-(l). Now we define four sets. Let W_C be the set of squares in C that does not intersect either of ℓ_v, ℓ_h; X_C be the set of squares in C that intersects only ℓ_v, but not ℓ_h; Y_C be the set of squares in C that intersects only ℓ_h, but not ℓ_v; and Z_C be the set of squares in C that intersects both ℓ_h and ℓ_v. Let W (resp. X, Y, Z) be the set of all squares of type W_C for all cells C in the grid decomposition. We select one of these sets W, X, Y, Z uniformly at random. Let $\mathcal{R}_3$ be the squares that survive. Then each square will survive with probability $\geq \frac{1}{8} \times \frac{1}{4} = \frac{1}{32}$.

Now we will look at the conflict graph H of $\mathcal{R}_3$ and prove that the independent sets of H are actually guillotine separable.

▶ **Lemma 27.** *The independent set of squares obtained from the conflict graph H as defined above, is guillotine separable*

Proof. Let P be a piece obtained from such sampling as defined above. We will show the existence of a guillotine cut that does not cut any of the squares in P. Iteratively, this will show guillotine separability of all rectangles embedded on P. If we only have one square in our piece, then we are done. So let us assume that we have at least two squares. Now consider the two squares with the lowest levels. Let the squares be R_1 and R_2 and the levels to which they belong are l_1 and l_2, respectively. There are two cases:

Case 1: $l_1 = l_2$. Then we have two subcases.

In *subcase (a)*, they belong to *different* grid cells. Then we can separate them by cutting along one of the grid lines coinciding with one of the edges of grid cells containing one of the two considered squares. This line does not cut any of the other squares because the level of this line is lesser than every square in the piece (by definition of R_1 and R_2). Now let us apply this cutting procedure as long as we have a set of squares having the lowest level and are in different grid cells. At the end, we should obtain a piece in which the lowest level squares are either of different levels or they belong to the same level and same cell.

In *subcase (b)*, they belong to the *same* grid cell. As sampling S gives at most one square per grid cell, R_1, R_2 belong to one of the groups among W, X or Y. But we can then separate any two squares from the same group along one of the level-l grid lines. This level-l line does not intersect any of the squares in this piece as we do not have any level-l squares outside this cell. By the first sampling, this line can not intersect any higher level squares.

Case 2: $l_1 \neq l_2$. Then we can cut along the grid line coinciding with the edges of grid cell of the second largest square. This does not cut any of the smaller squares by definition of the first sampling. Also this line does not intersect the largest square by the definition of independent set in conflict graph. ◀

Now we are ready to prove the final theorem of this section.

▶ **Theorem 28.** *Given a set of axis-parallel weighted squares embedded in a plane, there is always a guillotine cutting sequence that recovers* $\frac{1}{160}$*-fraction of weights.*

Proof. As the smallest square can have at most four neighbors after the first sampling step, inductively we can show the conflict graph to be 5-colorable. Hence, from Lemma 27 we can conclude that any set of weighted squares is ε-guillotine sampleable for $\varepsilon = 1/32 \times 1/5$. ◀

4 Extraction of rectangles

Using standard techniques from [1], we can assume that all rectangles are embedded in a $2n \times 2n$ rectangular box with all corners of rectangles having integral coordinates in $[2n \cup 0] \times [2n \cup 0]$. When we refer to the width or height of these rectangles we refer to the width or height of those rectangles in this embedding. W.l.o.g. assume that $\log n \in \mathbb{Z}$. Let us define some horizontal lines called *poles* and give an attribute to each rectangle called *level*. A set of *poles at level-i* is defined as equally spaced horizontal lines with y-coordinates $\{\frac{(2k+1)\times n}{2^{i-1}} \mid k \in [(2^{i-1}-1) \cup 0]\}$. A level-0 line has y-coordinate either 0 or $2n$. The *level of a rectangle* is defined as the smallest level i such that some pole at level i intersects the rectangle. The union of all poles from levels 1 to i divides the plane into 2^i equal partitions which we will call as the *grid-partition of level* i. Let $\mathcal{R}_i$ be the set of rectangles present in i'th level for $i \in [\log n \cup 0]$. Further, assume that $\alpha_i = |\mathcal{R}_i|$.

Now we will use poles to show that we can partition all the input rectangles into $\log n + 1$ groups such that the embedding of rectangles in each group is guillotine separable.

▶ **Theorem 29.** *Given a set of rectangles (possibly weighted)* $\mathcal{R}$ *embedded in a square,* $\frac{1}{\log n+1}$ *fraction of total rectangles (resp., weight) can be extracted using 2-stage cuts.*

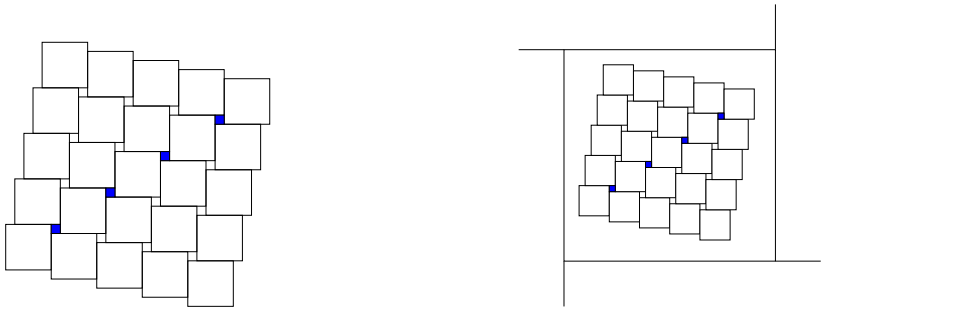
Proof. Rectangles in $\mathcal{R}_i$ are 2-stage separable, for any $i \in [\log n + 1]$. The first stage consists of cuts along the poles of level $\leq i-1$. These cuts divide the plain into 2^i equal partitions without cutting rectangles in $\mathcal{R}_i$. In each of the partitions, all rectangles are intersected by a pole of level-i and thus no vertical line within a partition can intersect two rectangles. Hence, the second stage can separate all rectangles by vertical cuts. As $\mathcal{R}_i$'s partition $\mathcal{R}$, taking $\mathcal{R}_i$ with the maximum cardinality (resp. weight) gives extraction factor of $\frac{1}{\log n+1}$. ◀

Using a k-ary partition, we can gain even further.

▶ **Lemma 30.** *We can extract* $n/\log_3(n+2)$ *rectangles by a series of guillotine cuts.*

Proof. We prove this by induction on n. The base case is trivial. Consider a T-cut partitioning the plane into three parts with almost equal number of rectangles on each part. Suppose the number of rectangles on the T is greater than $\frac{n}{\log_3(n+2)}$ then we can just extract all of it using Observation 20. Otherwise, by induction on each of the small parts we can extract $\frac{n-\frac{n}{\log_3(n+2)}}{\log_3(n+2-\frac{n}{\log_3(n+2)})-1} \geq \frac{n-\frac{n}{\log_3(n+2)}}{\log_3(n+2)-1} = n/\log_3(n+2)$. ◀

For general rectangles, we will show that we can not obtain much better extraction factor using $O(1)$-stage cuts. However, in Section 5, we use poling arguments to obtain $O(1)$-extraction factor for many special cases of rectangles.



(a) External square skew-grid. **(b)** Level-$(\alpha-1)$ instance in the hole of Level-α instance.

Figure 3 Level-α square skew-grid.

4.1 Hardness for k-stage algorithms in weighted case

In this section, we will prove Theorem 2.

▶ **Definition 31** ([1]). *A $k \times k$ unit square skew-grid is defined as a set of k^2 unit squares arraged in k rows and k columns numbered from bottom to top and left to right, respectively. With the bottom left coordinates of the square belonging to i^{th} row and j^{th} column as $(j + i \cdot \varepsilon, i - j \cdot \varepsilon)$ for $i, j \in [(k-1) \cup 0], \varepsilon \leq \frac{1}{k}$. A square in i^{th} row and j^{th} column is said to have* location *(i,j). The gaps formed by four adjacent squares are called holes. Those holes that are formed by the four squares with location $(i,i), (i+1, i+1), (i+1, i), (i, i+1)$ for $i \in [0 \leq [(k-2) \cup 0]$ are called diagonal holes and they are indexed by i.*

▶ **Observation 32.** *Let $\mathcal{S}$ be a $k \times k$ square skew-grid. Consider a set of $p \in [(k-1) \cup 0]$ vertical lines passing through different diagonal holes of $\mathcal{S}$ partitioning the plane into $p+1$ sub-pieces and in each of these pieces consider some horizontal lines through different diagonal holes (each hole has at most one line passing through it) with total number of horizontal lines $q \in [(k-1) \cup 0]$. This set of $p+q$ lines intersect a total of at least $k \cdot p + q$ squares.*

Proof. We can see that every vertical line through j^{th} diagonal hole intersects all squares with location $(i+1, j)$, $j \leq i$ and $(i, j+1)$, $i \leq j$. So all the p vertical lines intersect a total of $p \cdot k$ squares and none of these squares have location (i, i). Every horizontal line through j^{th} diagonal hole intersects the square with location (j, j), which gives a total of atleast q diagonal squares that get intersected by the horizontal lines. ◀

▶ **Definition 33.** *A level-α $k \times k$ square skew-grid (see Figure 3) is defined as a $k \times k$ square skew-grid with each of the $k-1$ diagonal holes having a level-$(\alpha-1)$ $k \times k$ skew-grid in it scaled appropriately to fit inside it. A level-0 square skew-grid is defined to be empty. We also say that a square belongs to level-i if it is among the largest squares in the level-i $k \times k$ square skew-grid that it is contained in.*

Now we show that for level-α $k \times k$ square skew-grid extraction factor is bounded based on the number of stages used. We assign weights uniformly to squares of same level so that the total weight of each level is 1. This implies the overall weight of this instance is α.

▶ **Theorem 34.** *Let $f_c(\alpha)$ be the best extraction factor for any c-stage extraction algorithm for level-α $k \times k$ square skew-grid instance. Then, $f_c(\alpha) \leq \frac{\alpha}{k} + c - 1$.*

Proof. We prove it by induction on $c+\alpha$. As we can extract at most one square out of k in each column by slicing vertically, we have $f_1(\alpha) \leq \alpha/k$ and $f_c(1) \leq 1$. Thus for $c=1$ and ($c \geq 2$ and $\alpha = 1$), the claim is true. This proves the base case: $c+\alpha = 2$. Let us prove the inductive step from $c+\alpha$ to $c+\alpha+1$. Let us assume that for every c, α with $c+\alpha \leq s$, the bound on $f_c(\alpha)$ is true. Now we will prove this for every $(c, \alpha+1)$ with $c+\alpha = s$.

Consider the best c-stage maximum weight extraction cutting sequence for a level-$(\alpha+1)$ instance. On every diagonal hole of it, a cutting sequence of stage $\leq c$ is induced. So considering an $(\alpha+1)$-stage configuration, suppose that the number of holes on which an i-stage cutting ($i \leq c$)) is induced is h_i. One observation is that the function $f_c(\alpha)$ is monotonic for a fixed α (extraction factor should be non decreasing by increasing stages). Thus using Observation 32,

$$f_c(\alpha+1) \leq \frac{k^2 - k \times h_c - h_{c-1}}{k^2} + \sum_{i=1}^{c} \frac{h_i}{k-1} \times f_i(\alpha) \tag{1}$$

$$\leq \frac{k^2 - k \times h_c - h_{c-1}}{k^2} + \sum_{i=1}^{c} \frac{h_i}{k-1} \times \left(\frac{\alpha}{k} + i - 1\right) \tag{2}$$

$$\leq 1 + h_c \times \left(\frac{\frac{\alpha}{k} + c - 1}{k-1} - \frac{1}{k}\right) + \sum_{i=1}^{c-1} h_i \times \left(\frac{\frac{\alpha}{k} + c - 2}{k-1} - \frac{1}{k^2}\right) \tag{3}$$

$$\leq 1 + h_c \times \left(\frac{\frac{\alpha}{k} + c - 1}{k-1} - \frac{1}{k}\right) + (k - 1 - h_c) \times \left(\frac{\frac{\alpha}{k} + c - 2}{k-1} - \frac{1}{k^2}\right) \tag{4}$$

$$\leq 1 + h_c \times \left(\frac{1}{k(k-1)} + \frac{1}{k^2}\right) + \frac{\alpha}{k} + c - 2 - \frac{k-1}{k^2} \tag{5}$$

$$\leq \frac{\alpha+1}{k} + c - 1 \tag{6}$$

The first term in RHS of (1) is the maximum total weight of level-α squares that can be extracted, following from Observation 32. The second summation term in (1) is the total weight extracted through the i^{th} stage cutting sequence induced on the diagonal holes over all $i \in [c]$. In (2) we can substitute $\frac{\alpha}{k} + i - 1$ in place of $f_i(\alpha)$ by inductive assumption. We get (3) from (2) by rearranging terms and replacing i for all $i \in [c-1]$ by $c-1$. Since $\sum_{i=1}^{c} h_i \leq k-1$ and also $\left(\frac{\frac{\alpha}{k}+c-2}{k-1} - \frac{1}{k^2}\right) \geq 0$, we can replace $\sum_{i=1}^{c-1} h_i$ by $k - 1 - h_c$ in (3). ◀

▶ **Observation 35.** *The total number of squares n in a level-α $k \times k$ square skew-grid is $k^2 \cdot \sum_{i=0}^{\alpha-1} (k-1)^i$. It follows that $(k-1)^{\alpha+1} \leq n \leq k^{\alpha+1}$.*

So it follows from the result that the maximum extraction fraction using any c-stage algorithm is $f_c(\alpha)/\alpha \leq 1/k + (c-1)/\alpha$ which can be bounded by $n^{-\frac{1}{\alpha+1}} + \frac{c-1}{\alpha}$, using Observation 35. Taking $\alpha = \frac{\log n}{\log\log n} - 1$, we get the total extraction factor as $\frac{1}{\log n} + \frac{(c-1)\cdot\log\log n}{\log n - \log\log n}$.

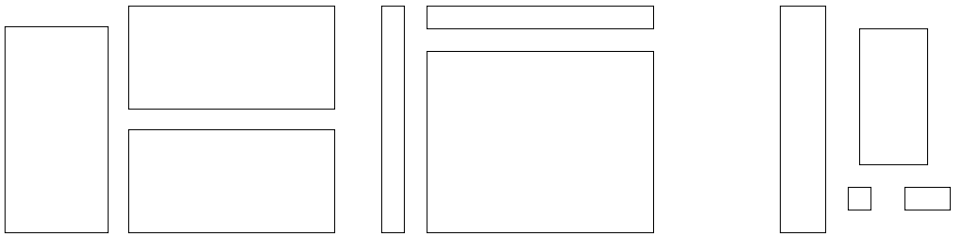
This concludes the proof of Theorem 2. Note that this does not disprove Conjecture 1 for $\Omega(n)$ stages as this instance already admits $O(1)$-extraction factor for $\Omega(n)$ stages.

For the unweighted case (Theorem 3), a more involved analysis shows that any guillotine cutting algorithm with a constant number of stages can extract at most $O(\frac{1}{\log\log n})$ fraction of total rectangles. See Section A.1 for the proof of Theorem 3.

We also consider guillotine separability of d-dimensional axis parallel disjoint hyper cubes. The proof of the following theorem is deferred to the full version.

▶ **Theorem 36.** *There exist a family of d-dimensional axis parallel disjoint hyper cubes for which the guillotine extraction factor is asymptotically upper bounded by $\frac{1}{2\cdot(d-1)}$.*

5 Constant extraction-factor for special classes



(a) fat rectangles ($\beta = 2$). **(b)** δ-large rectangles. **(c)** part-simillar instance ($\gamma = 2$).

Figure 4 special cases.

First, let us show a connection between the weighted case of guillotine separability of rectangles and MWISR (see Section B.1 for details).

▶ **Theorem 37.** *If there is a guillotine extraction algorithm which guarantees at least* $1/\alpha$ *fraction of the total weight, then we have a* $O(\alpha)$ *approximation algorithm for the MWISR problem that runs with running time* $O(n^5)$.

Now we show $\Omega(1)$-extraction factor for special classes of rectangles. Using Theorem 37, we obtain $O(1)$-approximation for MISR (also MWISR, except for the balanced instance) for these classes. For omitted proofs of this section, see Appendix B.

5.1 Fat rectangles

A rectangle i is fat if $\max_i(\frac{w_i}{h_i}, \frac{h_i}{w_i}) \leq \beta$, where β is a constant (see Figure 4a). We divide the rectangles into two sets: (i) $w \geq h$ and (ii) $w \leq h$, and pick the one with maximum weight. W.l.o.g assume this to be the former set. We generalize the grid sampling techniques for squares and randomly pick a size class modulo $(\log \beta + 2)$ among widths and modulo 2 among heights. Then we exploit properties of this sampling to extract a constant fraction.

▶ **Lemma 38.** *If a piece has* n *rectangles with* $\max_i(\frac{w_i}{h_i}, \frac{h_i}{w_i}) \leq \beta, \beta \geq 1$, *then there is a guillotine cutting strategy with extraction factor* $\frac{1}{1584 \cdot (\log \beta + 2)}$.

5.2 δ-large rectangles

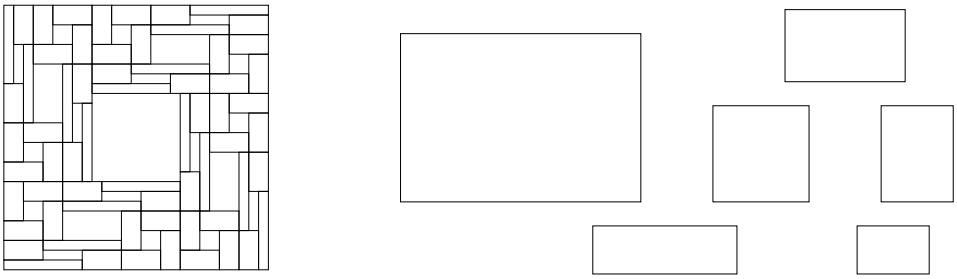
A rectangle i is δ-large if either $w_i \geq \delta N$ or $h_i \geq \delta N$, where $0 < \delta < 1$ (see Figure 4b). By using the poles that we have defined in Section 4, we obtain $O(1)$ extraction factor.

▶ **Lemma 39.** *Given an embedding of a set of* δ*-large rectangles in an* $N \times N$ *square, we can extract* $1/(\log(1/\delta) + 1)$ *fraction of rectangles (resp. weights) using 2-stage cuts.*

5.3 Part-similar instance

An instance is *part-similar* if $\min(\frac{\max_i h_i}{\min_i h_i}, \frac{\max_i w_i}{\min_i w_i}) = \gamma$, where γ is a constant, i.e., rectangles have either similar width or similar height (see Figure 4c). Using random position for poles, we find an interesting relation between the level of pole a rectangle belongs to and the size class of rectangle along the dimension perpendicular to the pole. This bounds the range of poles that any rectangle can belong to and we extract from one of the levels in that range.

▶ **Lemma 40.** *Given a set of rectangles with* $\min(\frac{\max_i h_i}{\min_i h_i}, \frac{\max_i w_i}{\min_i w_i}) = \gamma$, *there exists a 2-stage guillotine cutting sequence with extraction factor* $\frac{1}{2 \times (\lfloor \log \gamma \rfloor + 3)}$.



(a) balanced instance (every sub piece has a balanced $\sqrt{k}$ cut).

(b) anti-laminar instance (no rectangles is x-contained in another).

Figure 5 special cases.

5.4 Balanced instance

Balanced instances are instances where in any sub-piece with k rectangles, there exists a cut that cuts $c(k)$ rectangles ($c(k)$ should be either $\Omega(k)$ or $O(k^{1-\varepsilon})$, where $0 < \varepsilon < 1$), dividing the remaining rectangles in a balanced way such that the ratio of number of rectangles on both sides is at most a constant $r \geq 1$. We show a constant factor $O(1)$-extraction factor for these instances. This instance includes many probable candidates for worst-case examples, including the skew-grid (see Figure 5a).

▶ **Lemma 41.** *For a fixed $\alpha = 1 + \frac{1}{r}, r \geq 1, 0 \leq \varepsilon < 1, 0 < f$, if a configuration has the property that for every large sub-piece of k rectangles, there is a cut that cuts either at least $k \cdot f_1$ rectangles or at most $k^{1-\varepsilon} \cdot f$ rectangles and also divide rectangles into two sets such that the ratio of number of rectangles on the two sides is at most r, then we can extract $n \cdot f_1$ fraction of rectangles for $f_1 = e^{-\frac{f \cdot \alpha^{-\varepsilon}}{(1-f \cdot \alpha^{-\varepsilon})(1-\alpha^{-\varepsilon})}}, f \leq \alpha^{\varepsilon}$.*

5.5 Anti-laminar instance

A rectangle i is said to be x-contained (resp. y-contained) in another rectangle j if $x_j < x_i < x_i + w_i < x_j + w_j$ (resp. $y_j < y_i < y_i + h_i < y_j + h_j$). A set of rectangles is said to be x-containment free (resp. y-containment free) if no rectangle is x-contained (resp. y-contained) in other. A set of rectangles is *anti-laminar* if it is either x-containment free or y-containment free (see Figure 5b).

▶ **Lemma 42.** *For an anti-laminar instance, we always have an extraction factor $1/2$.*

6 Conclusion

We have made progress towards understanding guillotine separability of rectangles. We showed that Pach-Tardos conjecture is not true, even for squares, if we use $o(\log \log n)$-stages. However, if we allow $\Omega(n)$ stages, even with general rectangles we were unable to find any instance where we can not recover more than $n/2$ rectangles. The balanced instance (see Figure 5a) or its variants are probable candidates for such hard instances. However, we showed that we can still separate a constant fraction of rectangles from these instances. It is interesting to obtain guillotine separability of even $(\log \log n / \log n)$-fraction of rectangles. This will give an $O(n^5)$-time algorithms for MWISR, matching the present best approximation guarantee. Apart from the existential questions, an interesting problem is to find a polynomial-time $O(1)$-approximation algorithm to recover rectangles through a sequence of guillotine cuts.

References

1 Fidaa Abed, Parinya Chalermsook, José R Correa, Andreas Karrenbauer, Pablo Pérez-Lantero, José A Soto, and Andreas Wiese. On guillotine cutting sequences. In *APPROX*, pages 1–19, 2015.

2 Anna Adamaszek and Andreas Wiese. Approximation schemes for maximum weight independent set of rectangles. In *FOCS*, pages 400–409, 2013.

3 Nikhil Bansal, Jose R Correa, Claire Kenyon, and Maxim Sviridenko. Bin packing in multiple dimensions: inapproximability results and approximation schemes. *Mathematics of Operations Research*, 31:31–49, 2006.

4 Nikhil Bansal and Arindam Khan. Improved approximation algorithm for two-dimensional bin packing. In *SODA*, pages 13–25, 2014.

5 Nikhil Bansal, Andrea Lodi, and Maxim Sviridenko. A tale of two dimensional bin packing. In *FOCS*, pages 657–666, 2005.

6 Alberto Caprara. Packing 2-dimensional bins in harmony. In *FOCS*, pages 490–499, 2002.

7 Alberto Caprara, Andrea Lodi, and Michele Monaci. Fast approximation schemes for two-stage, two-dimensional bin packing. *Mathematics of Operations Research*, 30(1):150–172, 2005.

8 Parinya Chalermsook. Coloring and maximum independent set of rectangles. In *APPROX*, pages 123–134. Springer, 2011.

9 Henrik I Christensen, Arindam Khan, Sebastian Pokutta, and Prasad Tetali. Approximation and online algorithms for multidimensional bin packing: A survey. *Computer Science Review*, 24:63–79, 2017.

10 JM Valério de Carvalho and AJ Guimaraes Rodrigues. An lp-based approach to a two-stage cutting stock problem. *European Journal of Operational Research*, 84(3):580–589, 1995.

11 Waldo Gálvez, Fabrizio Grandoni, Afrouz Jabal Ameli, Klaus Jansen, Arindam Khan, and Malin Rau. A tight $(3/2+\varepsilon)$ approximation for skewed strip packing. *To appear in APPROX*, 2020.

12 Waldo Gálvez, Fabrizio Grandoni, Sandy Heydrich, Salvatore Ingala, Arindam Khan, and Andreas Wiese. Approximating geometric knapsack via l-packings. In *FOCS*, pages 260–271, 2017.

13 PC Gilmore and Ralph E Gomory. Multistage cutting stock problems of two and more dimensions. *Operations research*, 13(1):94–120, 1965.

14 Claire Kenyon and Eric Rémila. A near-optimal solution to a two-dimensional cutting stock problem. *Mathematics of Operations Research*, 25(4):645–656, 2000.

15 Arindam Khan. *Approximation algorithms for multidimensional bin packing*. PhD thesis, Georgia Institute of Technology, Atlanta, GA, USA, 2016.

16 Arindam Khan, Arnab Maiti, and Amatya Sharma. On guillotine separable packing of two-dimensional knapsack. *Manuscript*, 2020.

17 Joseph YT Leung, Tommy W Tam, CS Wong, Gilbert H Young, and Francis YL Chin. Packing squares into a square. *Journal of Parallel and Distributed Computing*, 10(3):271–275, 1990.

18 Sahil Mahajan. Guillotine cuts. Master's thesis, IIIT Delhi, 2016.

19 Michael L McHale and Roshan P Shah. Cutting the guillotine down to size. *PC AI*, 13:24–26, 1999.

20 János Pach and Gábor Tardos. Cutting glass. In *Proceedings of the 16th Annual Symposium on Computational Geometry*, pages 360–369. ACM, 2000.

21 Enrico Pietrobuoni. *Two-dimensional bin packing problem with guillotine restrictions*. PhD thesis, University of Bologna, Italy, 2015.

22 Jakob Puchinger, Günther R Raidl, and Gabriele Koller. Solving a real-world glass cutting problem. In *European Conference on Evolutionary Computation in Combinatorial Optimization*, pages 165–176. Springer, 2004.

23 W Schneider. Trim-loss minimization in a crepe-rubber mill; optimal solution versus heuristic in the 2 (3)-dimensional case. *European Journal of Operational Research*, 34(3):273–281, 1988.

24 Steven S Seiden and Gerhard J Woeginger. The two-dimensional cutting stock problem revisited. *Mathematical Programming*, 102(3):519–530, 2005.

25 Jorge Urrutia. Problem presented at accota'96: Combinatorial and computational aspects of optimization. *Topology, and Algebra, Taxco, Mexico*, 1996.

26 François Vanderbeck. A nested decomposition approach to a three-stage, two-dimensional cutting-stock problem. *Management Science*, 47(6):864–879, 2001.

27 PY Wang. Two algorithms for constrained two-dimensional cutting stock problems. *Operations Research*, 31(3):573–586, 1983.

A Omitted proofs from Section 4

A.1 Hardness for small stage algorithms in unweighted case

Here we prove Theorem 3.

▶ **Lemma 43** (Restatement of lemma 18 from [1]). *The number of squares of a $n \times n$ square skew-grid that can be seperated by a guillotine cutting strategy is at most $\lceil \frac{n^2+2n-2}{2} \rceil \leq \frac{n^2}{2} + n$*

In this subsection we show another example which is constructed almost similarly as in the weighted case such that the extraction factor is bounded based on the number of stages used. In the construction so that the number of squares in all levels is almost same.

▶ **Definition 44.** *For $k \geq 2$, A k-modified level-α square skew-grid is defined as a $\left(2^{2^{\alpha+k-1}}+1\right) \times \left(2^{2^{\alpha+k-1}}+1\right)$ square skew-grid with each of the $2^{2^{\alpha+k-1}}$ diagonal holes having a k-modified level-$(\alpha-1)$ square skew-grid in it scaled appropriately to fit inside it. A k-modified level-0 square skew-grid is defined to be empty. We also say that a square belongs to level-i if it is among the largest squares in the k-modified level-i square skew-grid that it is contained in.*

▶ **Lemma 45.** *Let $E_c(\alpha)$ be the maximum total number of squares that can be extracted using a c-stage cutting sequence on a level α configuration. Let $f_c(\alpha) = E_c(\alpha) \cdot 2^{-2^{k+\alpha}}$. Then $f_c(\alpha) \leq c - 1 + \sum_{i=1}^{\alpha}(2^{-2^{\alpha+k-i}} + 2^{-2^{\alpha+k+1-i}})$.*

Proof. We prove it by induction on $c+\alpha$. Let us prove some boundary cases first and use inductive step to cover all the cases. For the case when $c = 1$, it is clear that

$$E_1(\alpha) = (2^{2^{\alpha+k-1}}+1) + 2^{2^{\alpha+k-1}} \times (2^{2^{\alpha+k-2}}+1) + 2^{2^{\alpha+k-1}} \times 2^{2^{\alpha+k-2}} \times (2^{2^{\alpha+k-3}}+1) \ldots (7)$$

which gives $f_1(\alpha) = \sum_{i=1}^{\alpha}(2^{-2^{\alpha+k-i}} + 2^{-2^{\alpha+k+1-i}})$ which satisfies the bound we assumed. Now for the case when $c \geq 2$,$\alpha = 1$ we have

$$f_c(1) = E_c(1) \times 2^{-2^{1+k}} \leq 2^{-2^{1+k}} \times \left(\frac{(2^{2^k}+1)^2}{2} + 2^{2^k} + 1\right) \leq 1 \quad (8)$$

By using the upper bound given in Lemma 43 and the fact that $k \geq 2$, we get the above bound which also satisfies the bound for $f_c(\alpha)$ we assumed. We have proved the base case when $c + \alpha = 2$. Let us assume for every c, α with $c + \alpha \leq s$ the bound on $f_c(\alpha)$ is true. Let us prove that for every $(c, \alpha+1)$ with $c + \alpha = s$ the bound on $f_c(\alpha+1)$ is true (h_i is defined similarly as in the weighted case).

$$E_c(\alpha+1) \leq \left((2^{2^{\alpha+k}}+1)^2 - h_c \times (2^{2^{\alpha+k}}+1) - h_{c-1}\right) + \sum_{i=1}^{c} h_i \times E_i(\alpha) \quad (9)$$

$$\Rightarrow f_c(\alpha+1) \leq c - 1 + \sum_{i=1}^{\alpha+1}(2^{-2^{\alpha+k-i}} + 2^{-2^{\alpha+k+1-i}}) \quad (10)$$

The inequality (9) is obtained similarly as in the weighted case. We obtain (10) by using the inductive assumptions and some simple algebraic manipulations. ◀

Now we also have that the total number of squares $n = 2^{2^{\alpha+k}} \times \sum_{i=1}^{\alpha}(1+2^{-2^{k+i-1}})^2$. This gives the extraction fraction $E_c(\alpha)/n = f_c(\alpha)/\sum_{i=1}^{\alpha}(1+2^{-2^{k+i-1}})^2 \leq c-1+\sum_{i=1}^{\alpha}(2^{-2^{\alpha+k-i}}+2^{-2^{\alpha+k+1-i}})/\sum_{i=1}^{\alpha}(1+2^{-2^{k+i-1}})^2 \leq c/\alpha$ (because the numerator is $< c$ and denominator $> \alpha$). Now in order to get a good upper bound we need to minimize c/α for fixed n, c which means we need to maximize α for fixed n. which happens when $k = 2$. So $n \approx \alpha \cdot 2^{2^{\alpha}}$. Thus α is in the order of $\log\log n$ and the upper bound c/α in the order of $c/\log\log n$. Which says we need at least order $\log\log n$ number of stages for any algorithm to guarantee a constant extraction factor even for squares case. Strangely the results we got say that using constant stages we cannot guarantee better than $\frac{\log n}{\log\log n}$ and $\log\log n$ factors for the weighted and unweighted cases, respectively, which matches the best-known approximation ratios for MISR.

B Omitted proofs from Section 5

B.1 Connection between MWISR and weighted guillotine problem

In this subsection we prove Theorem 37. Note that Abed et al. [1] already showed a similar connection between MISR and the unweighted guillotine problem. We build on their approach to build a dynamic program to handle the weighted case.

We are given a set of n axis-parallel rectangles. We can assume that the corners of rectangles have integer coordinates in the range $\{0, \ldots, 2n\}$ and w.l.o.g no two rectangles are exactly coinciding (if such a case exists just consider the rectangle with the largest weight). Consider a piece P out of $O(n^4)$ such pieces possible in integer plane $[0, 2n] \times [0, 2n]$.
If P has no rectangle completely lying inside, we take the solution to be an empty set.
If $P = R$ (plane exactly coincides rectangle R), we take the maximum of the below two cases:
Case 1: Consider solution for P to be only R and discard all other rectangles inside R.
Case 2: Discard R and consider all rectangles inside R. Try all possibilities of dividing P into two smaller pieces using a horizontal or vertical guillotine cut such that the horizontal/vertical coordinates of this cut is an integer. Consider one such cut and let $P_1 \neq \emptyset \neq P_2$ denote the resulting pieces. The DP looks up the solutions for the cells representing P_1 and P_2 and combines them to a solution for P. It selects the cut yielding the optimal total profit from the resulting two subproblems. Let us define two tables $DPG[i][j][k][l]$ and $RECT[k_1][l_1][k_2][l_2]$. Where $DPG[i][j][k][l]$ stores the maximum weight guillotine separable independent set of rectangles in the piece having bottom left coordinates(k, l) and top right coordinates $(k+i, l+j)$ and $RECT[k_1][l_1][k_2][l_2]$ stores the weight of the rectangle in the input having bottom left and top right coordinates (k_1, l_1) and (k_2, l_2) respectively if such a rectangle is present else it stores zero otherwise.

$$DPG[i][j][k][l] = \max\Big(\max_{1\leq s\leq j-1}(DPG[i][s][k][l] + DPG[i][j-s][k+s][l]),$$
$$\max_{1\leq s\leq i-1}\big(DPG[s][j][k][l] + DPG[i-s][j][k+s][l]\big), RECT[k][l][k+i][l+j]\Big).$$

With constraints $1 \leq i \leq 2n$, $1 \leq j \leq 2n$, $0 \leq k \leq 2n-i$, $0 \leq l \leq 2n-j$.
This DP gives an $O(n^5)$ time algorithm and returns the maximum weight guillotine separable independent set of rectangles.

B.2 Fat rectangles

We give proof of Lemma 38. Let us consider objects that have bounded $\frac{w}{h} \leq \beta$, $\frac{h}{w} \leq \beta$, $\beta \geq 1$ for any rectangle in $\mathcal{R}$. Previously, a square of side length $\ell \in (\frac{N}{2^{i+1}}, \frac{N}{2^i}]$ is said to belong to level i. Now similarly, for a rectangle with width w and height h, we say that its width

w belongs to level i if $w \in (\frac{N}{2^{i+1}}, \frac{N}{2^i}]$ and height h belongs to level j if $h \in (\frac{N}{2^{j+1}}, \frac{N}{2^j}]$. Let us only consider that class of rectangles for which $h \geq w$ and we can assume w.l.o.g. that they are at least half in weight (resp. number). Now it follows that for every rectangle $j \leq i$, since $\beta \geq \frac{h}{w} > \frac{\frac{N}{2^{j+1}}}{\frac{N}{2^i}} = 2^{i-j-1}$, we have $j \leq i < j + 1 + \log\beta$. Now consider all the width classes and select all the classes that are congruent to some random $r \in [0, \log\beta + 1]$ modulo $(2 + \log\beta)$. Also randomly select either even or odd classes from the height class. The expected number(or weight) of rectangles that remain until now are $\frac{1}{4\cdot(2+\log\beta)}$. To this set of rectangles let us apply the sampling procedures that we applied for squares. In the first sampling, we remove all those rectangles that are intersected by vertical lines of level less than that of its width or horizontal lines of level less than that of its height. Which guarantees an expected number of $\frac{1}{4}$ number of rectangles that remained after our first two sampling procedures. In the next sampling procedure, we define a cell for a rectangle of level (i, j) (let us say for now informally that the level of a rectangle is (width class, height class) as the rectangle formed by the two pair of vertical and horizontal level $i-1$ and level $j-1$ lines that contain this rectangle. We want every cell of a rectangle to have at most one rectangle with level (i, j). It is an important observation that any rectangle with $i' < i$ or $j' < j$ cannot stay inside a level (i, j) cell, because $i' < i$ implies $i' \leq i - (2 + \log\beta) \leq i - 2$, which implies the width of this rectangle exceeds that of the cell. And $j' < j$ implies that $j' \leq j - 2$, which exceeds the height of the cell. A bug arises in the second sampling mentioned in [1] because a cell of level-i can have a level $i-1$ square inside it which causes problems in all the results that follow. Since there can be at most 9 rectangles of level (i, j) inside a level (i, j) cell. We select a square randomly. The total expected weight (resp. number) of squares that remain after the previous sampling is $\frac{1}{9}$. After these samplings we have an important property that we can use.

▶ **Observation 46.** *If a rectangle of level-(i', j') intersects one of the grid lines of a level (i, j) rectangle then $i' + j' < i + j$.*

Proof. Suppose the rectangle intersects the horizontal level $j-1$ line then we have $j' \leq j - 1 \leq i - 1$ and $i' < j' + \log\beta + 1 \leq i + \log\beta$, which implies $i' \leq i$, which then implies $i' + j' < i + j$. Suppose the rectangles intersects the vertical $i-1$ level lines then we have $i' \leq i - 1$, which implies $i' \leq i - 2 - \log\beta < j - 1$ and $j' \leq i' \leq i - 2 - \log\beta < j - 1$, which then implies $i' + j' < i + j$. We can also see that the total number of rectangles that can intersect the boundary of this cell can be at most 10 (4 in corners, 0 on vertical edges because $j' < j - 1$, 3 each on horizontal edges). ◀

We can define the conflict graph H same as defined for squares and it is easy to see that H is 11 colorable. And also if we think $i + j$ as the level of a rectangle now it also follows similarly that independent set of H is guillotine separable. So combining all of this gives a $\frac{1}{1584\cdot(\log\beta+2)}$ extraction algorithm. Even when we have bounded $\frac{w_{max}}{w_{min}}$ or $\frac{h_{max}}{h_{min}}$ we can extract $O(\frac{1}{\log\beta})$ where β is the bound using the same grid sampling techniques. However, we will solve that problem with a different technique in the later subsection.

B.3 δ-large rectangles

Now we prove Lemma 39.

Proof. All rectangles will be intersected by some poles of level $\leq \lceil\log(1/\delta)\rceil$. Similar to proof of Theorem 29, we can select the level with the maximum number of rectangles (resp. weights) to extract $n/(\log(1/\delta) + 1)$ rectangles (resp. weights) using 2-stages of cuts. ◀

B.4 Part-similar instance

Here we prove Lemma 40.

Proof. Let us define the size classes for a rectangles based on their height. A rectangle belongs to size class i if its height is in the range $(\frac{2n}{2^{i+1}}, \frac{2n}{2^i}]$ for $i \in [\log 2n \cup 0]$.

Now let us choose a random $y \in [2n \cup 0]$ uniformly and shift all the horizontal poles by y wrapping up appropriately. Let us delete all those rectangles that are intersected by a pole of level less than its size class. The probability that a rectangle stays after this sampling is at least $1/2$. Which implies that there exists a y such that more than half of the rectangles are not intersected by a line with level lesser than their size classes. Now if an object has height $> \frac{N}{2^{i+1}}$ then it has to belong to level $\leq i+1$. This implies that after the random sampling, every rectangle from size class i belongs to level i or $i+1$. Suppose we have a set of rectangles with $\frac{h_{max}}{h_{min}} = \gamma$. Let all the rectangles after the sampling be distributed over levels $[\alpha, \alpha + c - 1]$. The minimum size difference between any two rectangles in levels α and $\alpha + c - 1$ is $> 2^{c-3}$. So $\gamma > 2^{c-3}$ gives $c \leq \lfloor \log \gamma \rfloor + 3$. This guarantees a level with at least $\frac{n}{2\times(\lfloor \log \gamma \rfloor + 3)}$ rectangles. ◀

B.5 Balanced instance

▶ **Lemma 47.** *For an instance of n rectangles, if in any sub-piece with k rectangles we have a cut that cuts at most $c(k)$ rectangles dividing the remaining rectangles in a balanced way such that the number of rectangles on both sides has ratio at most $r \geq 1$, then we have an extraction factor $f(n) \geq n \cdot e^{-\frac{\alpha}{\alpha - c(\alpha)} \cdot \sum_{i=1}^{\log_\alpha n} \frac{c(\alpha^i)}{\alpha^i}}$, where $\alpha = 1 + \frac{1}{r}$ (note that in this result $c(k)$ can be any function).*

Let us start with a simpler case where any sub-configuration obtained after few cuts has a $O(\sqrt{p})$ cut that divides the number of rectangles on each side almost equally, where p is the number of rectangles in that sub-configuration. Let $f(n) = n/g(n)$ be the number of rectangles that can be extracted by simply cutting using the available $O(\sqrt{n})$ cut. We have $f(n) \geq 2 \cdot f((n - \sqrt{n})/2)$. This implies $g(n) \leq (\frac{n}{n-\sqrt{n}}) \cdot g(\frac{n-\sqrt{n}}{2}) \leq (\frac{n}{n-\sqrt{n}}) \cdot g(\frac{n}{2})$ (since $g(n)$ is a non decreasing function). We have base cases $g(p) = 1, p \leq 3$. Using the above inequality recursively, we obtain the upper bound for $g(2^k)$ as $\prod_{i=1}^{k} (1 - 2^{-i/2})^{-1} = e^{-\sum_{i=1}^{k} \log(1-2^{-i/2})}$. Using $1 - 1/x \leq \log(x)$ for $x \geq 0$, we obtain $g(2^k) \leq e^{\sum_{i=1}^{k} \frac{2^{-i/2}}{1-2^{-i/2}}} \leq e^{\sum_{i=1}^{k} \frac{2^{-i/2}}{1-2^{-1/2}}} \leq e^{3\sqrt{2}+4}$. So $f(n) \geq n/e^{3\sqrt{2}+4}$.

Now we state the proof of Lemma 47, a more general case.

Proof. In balanced instance, in any sub-configuration with k rectangles we always have a cut which cuts at most $c(k)$ and divides the remaining rectangles in a balanced way such the number of rectangles on both sides are in some $r : 1$ ratio with $r \geq 1$. Then we have $f(n) \geq f(\frac{r}{r+1} \cdot (n - c(n))) + f(\frac{1}{r+1} \cdot (c - c(n)))$. This after replacing $f(n)$ with $n/g(n)$ gives $\frac{n}{g(n)} \geq \frac{(\frac{r}{r+1}) \cdot (n - c(n))}{g(\frac{r}{r+1} \cdot (n - c(n)))} + \frac{\frac{1}{r+1} \cdot (n - c(n))}{g(\frac{1}{r+1} \cdot (n - c(n)))}$. Now using the fact that the function $g(n)$ is non decreasing, we obtain $g(n) \leq (1 - \frac{c(n)}{n})^{-1} \cdot g(\frac{n}{1+\frac{1}{r}})$. Taking $\alpha = 1 + 1/r$ and applying the inequality recursively, we get that $g(n) \leq e^{-\frac{\alpha}{\alpha - c(\alpha)} \cdot \sum_{i=1}^{\log_\alpha n} \frac{c(\alpha^i)}{\alpha^i}}$. Whenever $c(n) = O(n^{1-\varepsilon})$ and $\alpha = 1 + \frac{1}{r}$ is constant, the value of $g(n)$ is bounded by a constant. For $c(n) = n^{1-\varepsilon}$ we get $f(n) \geq n \cdot e^{-\frac{\alpha^\varepsilon}{(\alpha^\varepsilon - 1)^2}}$. This implies we can always extract a constant fraction of the number of rectangles. ◀

The proof of Lemma 41 follows from the proof of Lemma 47. Using Lemma 47, we can extract a constant fraction of items if $c(k)$ is either $\Omega(k)$ or $O(k^{1-\varepsilon})$. The idea is that if $c(k)$ is $\Omega(k)$ then we can extract all the rectangles that are cut and this is already a constant fraction. If $c(k)$ is $O(k^{1-\varepsilon})$ then we can cut and recurse into the subpieces.

B.6 Anti-laminar instance

Intuitively, a set of rectangles is *anti-laminar*, if along one of the axes, the intervals formed by projecting the rectangles on that axis have the property that no interval is completely inside another. First we show the proof of Lemma 48.

▶ **Lemma 48.** *Let I be a set of n intervals on the x-axis such that no two intervals are completely contained inside each other. Let $\{p_1, \cdots, p_{2n}\}$ be the endpoints of intervals in I. Then we can distribute weight n to the intervals (p_i, p_{i+1}) for $i \in [2n-1]$ such that each interval in I gets weight 1.*

Proof. Let us prove this by induction on the number of intervals. This is obviously true when there is only one interval. Let us sort the intervals by increasing value of the left end point and label them $1, \ldots, k+1$. Suppose the claim is true until $n = k$. Then for $n = k+1$ we can assume that the first interval intersects the second interval w.l.o.g (otherwise we can assign weights for intervals other than the first interval and assign unit weight to the first interval which completes the inductive step) and consider the left end point of second interval. Consider the k intervals to the right of this point and assign weights so that all this k intervals have the same weights. Now since the second interval is not contained in the first interval, the right end point of the second interval is to the right of the right end point of the first interval. We can just assign the weight contained in the region from the right end point of first interval to the right end point of the second interval in the region from the left end point of the first interval to the left endpoint of the second interval. Which makes the weight of first interval equal to that of the second and hence this gives equal weight to all the intervals. We can also observe that this is impossible when an interval is contained in another interval. ◀

▶ **Lemma 49** (Restatement of Lemma 13 in [1])**.** *For a set of rectangles with equal width (or height) there exists a guillotine cutting strategy that separates at least $1/2$ of the weight.*

Now we are ready to prove of Lemma 42.

Proof. Without loss of generality let us suppose that the projection of rectangles on x-axis satisfies the no containment property. By Lemma 48, we can assign weights to intervals on x-axis such that the total weight inside each interval is same. Now if we scale each strip associated with the intervals with a scaling factor equal to its weight then we get another set of rectangles where each rectangle has the same width. Now by lemma 49 we can separate $1/2$ the total weight of the given set of rectangles. ◀

Maximizing Throughput in Flow Shop Real-Time Scheduling

Lior Ben Yamin
Computer Science Department, Technion, Haifa, Israel
lior.b@cs.technion.ac.il

Jing Li
Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA
jingli@njit.edu

Kanthi Sarpatwar
IBM T. J. Watson Research Center, Yorktown Heights, NY, USA
sarpatwa@us.ibm.com

Baruch Schieber
Department of Computer Science, New Jersey Institute of Technology, Newark, NJ, USA
sbar@njit.edu

Hadas Shachnai
Computer Science Department, Technion, Haifa, Israel
hadas@cs.technion.ac.il

Abstract

We consider scheduling real-time jobs in the classic flow shop model. The input is a set of n jobs, each consisting of m segments to be processed on m machines in the specified order, such that segment I_i of a job can start processing on machine M_i only after segment I_{i-1} of the same job completed processing on machine M_{i-1}, for $2 \leq i \leq m$. Each job also has a release time, a due date, and a weight. The objective is to maximize the throughput (or, profit) of the n jobs, i.e., to find a subset of the jobs that have the maximum total weight and can complete processing on the m machines within their time windows. This problem has numerous real-life applications ranging from manufacturing to cloud and embedded computing platforms, already in the special case where $m = 2$. Previous work in the flow shop model has focused on makespan, flow time, or tardiness objectives. However, little is known for the flow shop model in the real-time setting. In this work, we give the first nontrivial results for this problem and present a pseudo-polynomial time $(2m + 1)$-approximation algorithm for the problem on $m \geq 2$ machines, where m is a constant. This ratio is essentially tight due to a hardness result of $\Omega(\frac{m}{\log m})$ for the approximation ratio. We further give a polynomial-time algorithm for the two-machine case, with an approximation ratio of $(9 + \varepsilon)$ where $\varepsilon = O(1/n)$. We obtain better bounds for some restricted subclasses of inputs with two machines. To the best of our knowledge, this fundamental problem of throughput maximization in the flow shop scheduling model is studied here for the first time.

2012 ACM Subject Classification Mathematics of computing → Combinatorial optimization; Theory of computation → Scheduling algorithms

Keywords and phrases Flow shop, real-time scheduling, throughput maximization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.48

Category APPROX

Funding *Jing Li*: Supported by NSF CNS-1948457.



© Lior Ben Yamin, Jing Li, Kanthi Sarpatwar, Baruch Schieber, and Hadas Shachnai;
licensed under Creative Commons License CC-BY



1 Introduction

Flow shop is a fundamental scheduling model where a set of jobs needs to be processed in multiple stages in a specified order that is the same for all jobs. There are many real-world applications of flow shop scheduling, ranging from production planning and computing platforms to satellite systems and service centers. For instance, an autonomous car runs applications, such as obstacle detection and route planning, applying deep neural networks on an embedded computing platform, which are composed of a CPU host and a GPU accelerator connected via a non-preemptive bidirectional bus. Each execution instance (i.e., job) of the applications is first initiated in the CPU to preprocess the input data, then transfers the data from CPU to GPU via the bus, executes the computation on the GPU, and finally transfers the results back via the bidirectional bus. There are multiple such jobs running in real-time with different release times and deadlines, e.g., multiple images to be processed by the object detection application in a time window. Hence, the computing platform needs to schedule the execution on the CPU and GPU, as well as the data transfers on the bus, to meet preset deadlines, e.g., to maximize the number of images processed in a time window.

The flow shop model has been widely studied for minimizing the latest completion time of any job (or, *makespan*) since the 1950s, starting with the seminal work of Johnson [16], which showed that makespan minimization in two-machine flow shop can be solved in polynomial time. However, most extensions of the problem are strongly NP-hard [7]. For example, makespan minimization for flow shop with three machines is already NP-complete, even if the input length is measured by the sum of the job lengths [12]. Hence, later works studied approximation algorithms for the problem (see, e.g., [13, 18, 20, 22, 24]).

In this paper, we are interested in flow shop scheduling for jobs with different release times, due dates, and weights, and the scheduling objective is to maximize the throughput – the total weight of the jobs that are completed by their due dates. Surprisingly, in contrast to the extensive results on minimizing the makespan, flow time, and tardiness in the flow shop model, there is little work on maximizing the throughput of jobs with due dates. On the other hand, the problem of maximizing the throughput of jobs with release times, due dates, and weights, also known in the literature as (aperiodic) *real-time scheduling*, has been widely studied. In this classic model, each job can be processed to completion on a *single* machine or any of the *parallel* machines (see, e.g., [1–4, 9, 15, 17, 19, 25]).

We now formalize our problem. In the m-machine flow shop model, there is a set of n jobs, $\mathcal{J} = \{J_1, \ldots, J_n\}$, and m machines, $M_1, \ldots, M_m$. Each job J_j, $1 \leq j \leq n$, has a release time, a due date, and a weight, given by $r_j \geq 0$, $d_j \geq 0$, and $w_j > 0$, respectively. A job can only start executing on machine M_i, $2 \leq i \leq m$, after it has finished its execution on the previous machine M_{i-1}. In addition, at any time $t \geq 0$, each of the machines can process at most one job. For a job J_j, we denote the processing time of its i-th segment to be executed on machine M_i by $p_{j,i}$. We assume that $p_{j,i}, r_j$ and d_j are rational numbers. We further assume that all job segments are non-preemptive. In other words, once a job J_j has started its execution on a machine, this machine cannot stop or switch to another job until J_j has finished its execution on this machine. We seek a subset of jobs $\mathcal{J}' \subseteq \mathcal{J}$ that can be *feasibly* scheduled (i.e., each job J_j can complete processing on all machines in a flow shop manner in its time window $(r_j, d_j]$) and has a maximum total weight. We denote this *maximum throughput* objective by MaxT. We obtain results for MaxT in the m-machine flow shop model, with a focus on the special case of two-machine flow shop.

1.1 Applications

In the following, we motivate MaxT in the flow shop model with some real-life applications.

Scheduling in Cloud Data Centers. A *cloud-data-center (CDC)* consists of a set of server clusters connected with clients through a network. Since all the resources are stored on the servers, clients generate resource requests from the CDC. A data request consists of two steps: task *execution*, in which data is obtained from a disk or distributed storage systems and stored in memory, and then *transmission* from memory to the client over the network (see, e.g., [26]). When data requests have release times and due dates, a natural goal is to maximize the total number of requests that can be processed by the CDC in a given time interval. This yields an instance of MaxT in the flow shop model.

Earth Observation Satellites. An earth observation satellite (EOS) is equipped with high-resolution cameras for observing target objects across the surface of Earth. There are available time windows for multiple EOSs to observe a given object and to download the acquired image/video data to ground receiver stations. The problem of observation scheduling (at stage 1) and data downlink scheduling (at stage 2) with the objective of maximizing the number of satellites that can complete processing in their time windows yields an instance of MaxT in the two-machine flow shop model (see, e.g., [26]).

Autonomous Vehicle Navigation. An object detection application running in autonomous cars takes images from a front-facing camera as input and produces car steering angles as output (see, e.g., [6]). Since the algorithm uses deep neural networks (DNN), each image is handled in stages (preprocessing data in CPU, data transfers between CPU and GPU, and DNN computation in GPU), the process of handling the images in real-time so as to maximize the number of images processed in a given time window can be viewed as a MaxT instance in the flow shop model.

1.2 Contributions and Techniques

We say that $\mathcal{A}$ is a ρ-approximation algorithm for a maximization problem Π, for $\rho \geq 1$, if for any instance I of Π, $\mathcal{A}(I) \geq \frac{OPT(I)}{\rho}$, where $OPT(I)$ is the value of an optimal solution for I.

In this paper, we study the fundamental problem of throughput maximization in the flow shop scheduling model. Our main result is a polynomial-time $(9+\varepsilon)$-approximation algorithm for MaxT in the two-machine flow shop, where $\varepsilon = O(1/n)$ for an input of size n (i.e., $n = |\mathcal{J}|$). We derive the algorithm by first obtaining a pseudo-polynomial time $(2m+1)$-approximation algorithm for MaxT on m machines, where $m \geq 2$ is a constant. We note that the ratio of $(2m+1)$ is essentially tight for any $m \geq 3$, due to a known hardness of approximation result for a ratio $\Omega(\frac{m}{\log m})$ [14].

We show that MaxT admits better approximations on some restricted instances of the two-machine model. In particular, we present a 4-approximation algorithm for instances where all jobs have the same release time, i.e., $r_j = 0$ for all $J_j \in \mathcal{J}$, and uniform weights. For the special case where all jobs have the same time window and arbitrary weights, we give a $(3+\varepsilon)$-approximation algorithm, for any fixed $\varepsilon > 0$.

Techniques. In Section 2, we give an approximation algorithm for instances of MaxT on m machines, where m is some constant. As our algorithm requires solving a Configuration Linear Program (LP), this implies a pseudo-polynomial running time. Showing that this

algorithm can be implemented in polynomial time, with only a slight degradation in the approximation ratio, is a major challenge even in the two machine case. We use the following key observation. Any instance $\mathcal{J}$ can be modified to an instance $\mathcal{J}_{new}$ (by replacing some jobs with *new* jobs) in which for every job $J_j \in \mathcal{J}_{new}$ either *both* $p_{j,1}$ and $p_{j,2}$ are *large* relative to $d_j - r_j$, or *both* $p_{j,1}$ and $p_{j,2}$ are *small* relative to $d_j - r_j$. Then, by an intricate analysis, we show how to reduce the number of variables associated with the jobs in $\mathcal{J}_{new}$ to be of size polynomial in $|\mathcal{J}_{new}|$ (and consequently also in $|\mathcal{J}|$, since we add only a polynomial number of new jobs) with only a minor degradation in the quality of the solution. The resulting polynomial-size linear program can then be solved and rounded in polynomial time to obtain an approximate solution (details are in Section 3). In one of the special cases, we establish a precise relation between the approximability of classic real-time scheduling on a single machine and MaxT in the two-machine flow shop (details are in Section 4.1). This allows the use of approximation algorithms for the single machine case for solving our problem.

1.3 Prior Work

The problem of real-time scheduling with the objective of throughput maximization is discussed widely in the literature. A general instance of the problem consists of a set of n jobs and k machines, for some $k \geq 1$, where each job j has a weight $w_j > 0$, a release time r_j, a due date d_j and a processing time p_{ji} on machine i, for $1 \leq i \leq k$ and $1 \leq j \leq n$. The goal is to find a non-preemptive schedule that maximizes the weight of jobs that meet their respective due dates. Note that all the related works in this domain do not consider the flow shop model. Instead, the k machines form a single stage, where each job needs to be processed on *any one* of the machines.

The problem is known to be NP-complete already in the single machine case (i.e., $k = 1$), where all jobs have the same (unit) weight [11]. Some special cases of the problem are known to be solvable in polynomial time. Moore [19] showed for the single machine case and uniform job weights that, if $r_j = 0\ \forall j$ the problem can be solved in time $O(n^2)$. Sahni presented in [21] a *fully polynomial time approximation scheme (FPTAS)*, whose running time is $O(\frac{n^3}{\varepsilon})$, for the more general case where jobs have the same release time and arbitrary weights.

Bar-Noy et al. [3, 4] considered the real-time scheduling problem for general instances with k machines, for some $k \geq 1$, where jobs may have arbitrary weights and arbitrary release times and due dates. They presented in [3] a $(2+\varepsilon)$-approximation algorithm, using the local ratio technique. A quasi-polynomial time dynamic programming framework was proposed in [15], which gives a $(1+\varepsilon)$-speed $(1+\varepsilon)$-approximation algorithm for the weighted throughput problem on k machines. The best known result without speed augmentation is an $\frac{e}{e-1} < 1.582$ approximation algorithm [9], for a single machine and uniform job weights.

We note that a variant of MaxT, where for every job $J_j \in \mathcal{J}$, the start-times of all segments of J_j are given explicitly, yields an instance of *maximum weight independent set* in m-union graphs. Recall that an m-union graph can be modeled as the intersection graph of m-segments, i.e., each vertex in the graph can be represented by at most m segments on the real line. Two vertices are adjacent if their ith segments intersect. For this problem, the paper [5] presented a $2m$-approximation algorithm that was shown to be close to the best possible, due to a hardness of approximation result for a ratio of $\Omega(\frac{m}{\log m})$ [14]. The hardness result carries over to MaxT in the flow shop model and m machines, for any $m \geq 3$. Our approximation algorithm for a constant number of machines builds on an algorithm presented in [5]. However, further steps are required to obtain polynomial running time, which is the main contribution of this paper.

Another line of work that relates to MaxT deals with maximizing the total weight of *just-in-time (JIT)* jobs, i.e., the weighted number of jobs that are completed *exactly* on their due dates. All previous studies assume that $r_j = 0\ \forall j$. Choi and Yoon [8] show that JIT two-machine flow shop with arbitrary job weights is NP-complete. The special case of uniform job weights is solvable in polynomial time on two machines and is strongly NP-hard for instances with three machines. The best known result is an FPTAS in [10] (see also [23]).

We are not aware of earlier studies of throughput maximization in the flow shop model.

2 Approximation Algorithm for Fixed Number of Machines

In this section, we present a pseudo-polynomial time algorithm for MaxT on flow shop instances with $m \geq 2$ machines, where m is some constant. Given the set of jobs $\mathcal{J}$, each job J_j, $1 \leq j \leq n$, is associated with m segments and a weight $w_j \geq 0$. Also, J_j has a release time and a due date, $r_j \geq 0$ and $d_j \geq 0$, respectively. We seek a subset of the jobs that can be feasibly scheduled on the machines in a flow shop manner, such that the total weight of scheduled jobs is maximized.

As the processing time $p_{j,i}$ on machine M_i, release time r_j, and deadline d_j of job J_j are all rational numbers, we can obtain integer values for these parameters by appropriate scaling. Since all these values are integral, it is easy to see that any feasible solution can be "tweaked" so that the start times of all segments of all jobs begin at an integral time point. Thus, from now on we assume that this is the case. This allows us to discretize the input and consider all the possible occurrences of a job J_j in its time window $(r_j, d_j]$. An occurrence of J_j specifies the start times of all segments of J_j on the m machines in $(r_j, d_j]$. Note that the number of such possible occurrences of job J_j is upper bounded by $(d_j - r_j)^m$.

We give some notations towards solving MaxT on m machines using a linear program. Let $\mathcal{L}_j$ denote the set of occurrences of job J_j, so the number of the occurrences of J_j is $|\mathcal{L}_j|$. Let $\mathcal{L} = \bigcup_{j=1}^{n} \mathcal{L}_j$. Clearly, $|\mathcal{L}| = \sum_{j=1}^{n} |\mathcal{L}_j|$. Let $x^\ell(j) \in \{0, 1\}$ be an indicator variable for the selection of the ℓ-th occurrence J_j^ℓ of J_j in the solution, where $1 \leq \ell \leq |\mathcal{L}_j|$. We note that the number of variables and the number constraints in the linear program is $O(\sum_{j=1}^{n}(d_j - r_j)^m)$, and is thus pseudo-polynomial in the input size. Let $\mathbf{w} \in \mathbb{R}^n$, $\mathbf{x} \in \mathbb{R}^{|\mathcal{L}|}$ be a weight vector and a relaxed indicator vector, respectively. Then, $\mathbf{w} \cdot \mathbf{x} = \sum_{j=1}^{n} \sum_{\ell=1}^{|\mathcal{L}_j|} w_j x^\ell(j)$.

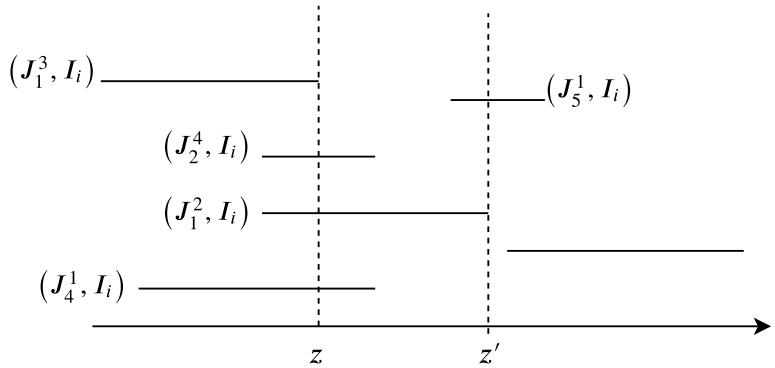


Figure 1 The job clique $(J_1^3, J_2^4, J_1^2, J_4^1)$ is defined by z, the right endpoint of the ith segment of J_1^3. The clique contains two occurrences of J_1: J_1^3 and J_1^2. The next clique, (J_1^2, J_5^1), is defined by the segment having its right endpoint at z'.

We now define *job cliques* on each of the m machines as follows. For machine $1 \leq i \leq m$, we examine the time axis from left to right and find among the segments that need to be processed on machine i a segment whose right endpoint is earliest. Let z be the time point

where this segment ends. We now define a clique $\mathcal{C}$ consisting of all job occurrences whose ith segment intersects the time point z. The next clique is defined by the earliest endpoint z' of an ith segment of a job, for which the following holds: there exists a job occurrence J_k^r, such that the ith segment of J_k^r intersects z', but $J_k^r \notin \mathcal{C}$, as shown in Figure 1. Intuitively, the endpoints z and z' in the definition of job cliques capture the maximum intersecting job occurrences in the time window $[z, z']$. Hence, a feasible schedule can only select one job occurrence in each clique.

We formulate the linear programming relaxation for MaxT as follows.

$$
\begin{array}{llll}
\text{(P)} \quad \text{maximize} \quad \mathbf{w} \cdot \mathbf{x} \quad \textit{subject to}: & & & \\
\displaystyle\sum_{J_j^\ell \in \mathcal{C}} x^\ell(j) & \leq & 1 & \text{for each clique } \mathcal{C} \\
\displaystyle\sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(j) & \leq & 1 & \forall 1 \leq j \leq n \\
x^\ell(j) & \geq & 0 & \forall 1 \leq j \leq n,\ 1 \leq \ell \leq |\mathcal{L}_j|
\end{array}
$$

The first constraint ensures that at most one job occurrence is selected from each clique. The second constraint guarantees that at most one occurrence of a job J_j is selected for the solution, $\forall J_j \in \mathcal{J}$. We note that (P) can be viewed as a Configuration LP, where each occurrence, J_j^ℓ, defines a *configuration*, $\forall J_j \in \mathcal{J}$.

Considering the neighbors of a job occurrence J_j^ℓ, we define two subsets of job occurrences.

1. Let $\tilde{N}_1(J_j^\ell)$ be the set of all job occurrences J_k^r where $k \neq j$, such that a segment of J_k^r intersects a segment of J_j^ℓ (recall that two segments can intersect only if both need to be processed on the *same* machine).
2. Let $\tilde{N}_2(J_j^\ell)$ be the set of all job occurrences J_j^r where $r \neq \ell$, i.e., other occurrences of J_j.

The *neighborhood* of a job occurrence J_j^ℓ is defined as $\tilde{N}(J_j^\ell) = \tilde{N}_1(J_j^\ell) \bigcup \tilde{N}_2(J_j^\ell)$.

▶ **Lemma 1.** *Let* $\mathbf{x}$ *be a feasible solution to* (P). *There exists a job occurrence* J_j^ℓ *satisfying:*

$$x^\ell(j) + \sum_{J_k^r \in \tilde{N}(J_j^\ell)} x^r(k) \leq 2m + 1.$$

Proof. We first show that there exists a job occurrence J_j^ℓ for which the following holds:

$$x^\ell(j) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} x^r(k) \leq 2m. \tag{1}$$

For two "neighboring" job occurrences J_j^ℓ and J_k^r (i.e., $J_k^r \in \tilde{N}_1(J_j^\ell)$ and $J_j^\ell \in \tilde{N}_1(J_k^r)$), define $z(J_j^\ell, J_k^r) = x^\ell(j) \cdot x^r(k)$. We also define $z(J_j^\ell, J_j^\ell) = (x^\ell(j))^2$. We prove (1) using a weighted averaging argument, where the weights are the values $z(J_j^\ell, J_k^r)$ for all pairs of job occurrences which have intersecting segments. The full proof is given in Appendix A. ◀

2.1 The Algorithm

We now show how to use Lamma 1 to get a $(2m+1)$-approximation for MaxT on m machines for some constant $m \geq 2$. Let $\mathcal{I}$ be the set of all half open subintervals of the interval $(0, 2m+1]$. Given an optimal solution $\mathbf{x}$ for the linear program (P), we construct a mapping $\psi : \mathcal{L} \to 2^{\mathcal{I}}$ such that for each job occurrence J_j^ℓ the following properties are satisfied:

1. All the subintervals in $\psi(J_j^\ell)$ are disjoint.
2. The total size of the subintervals in $\psi(J_j^\ell)$ is $x^\ell(j)$.
3. None of the subintervals in $\psi(J_j^\ell)$ intersects any of the subintervals in $\bigcup_{J_k^r \in \tilde{N}[J_j^\ell]} \psi(J_k^r)$.

The mapping is constructed for one job occurrence at a time according to a *hierarchical* order induced by Lemma 1. We first define this hierarchical order. The *last* job occurrence in the order is the occurrence J_j^ℓ that satisfies the inequality of Lemma 1. We then remove this job occurrence from the feasible solution to (P). We still remain with a feasible solution to (P) and we can apply Lemma 1 again and find yet another job occurrence that satisfies the inequality of the lemma. We append this job occurrence to the order. We continue in the same manner until we order all the job occurrences.

We compute $\psi(J_j^\ell)$ for one job occurrence at a time from the first to the last in the hierarchical order defined above. When $\psi(J_j^\ell)$ is computed, we remove from the interval $(0, 2m+1]$ all the subintervals in $\bigcup_{J_k^r \in \bar{N}} \psi(J_k^r)$, where $\bar{N} \subset \tilde{N}[J_j^\ell]$ is the set of all job occurrences in $\tilde{N}[J_j^\ell]$ that precede J_j^ℓ in the hierarchical order. By Lemma 1 the total size of these subintervals is no more than $2m + 1 - x^\ell(j)$. Thus, the remainder contains a set of disjoint subintervals of a total size at least $x^\ell(j)$. If we assign $\psi(J_j^\ell)$ greedily, that is, we assign the leftmost collection of such disjoint subintervals, then it can be shown that $|\psi(J_j^\ell)|$ is bounded by $|\mathcal{L}|$. This is because each job occurrence may increase the number of subintervals by at most one.

For a point $y \in (0, 2m+1]$, let $\phi(y) \subseteq \mathcal{L}$ be the subset of $\mathcal{L}$ consisting of all job occurrences J_j^ℓ for which one of the subintervals in $\psi(J_j^\ell)$ contains the point y. From the definition of the mapping ψ, it is evident that the subset $\phi(y)$ does not contain two job occurrences that intersect and also does not contain two job occurrences of the same job. Thus, the job occurrences in $\phi(y)$ can be scheduled feasibly to yield a weight of $w(y) = \sum_{J_j^\ell \in \phi(y)} w_j$. Let $y^* = \arg\max_{y \in (0,2m+1]}\{w(y)\}$. Note that if the mapping is computed greedily there are at most $|\mathcal{L}|^2$ possible values of $w(y^*)$. These values are determined by the right endpoints of all subintervals. The pseudocode of the algorithm, Flowshop_Time_Windows, is in Algorithm 1.

Algorithm 1 Flowshop_Time_Windows.

Find an optimal solution $\mathbf{x}$ for the linear program (P).
Order the job occurrences according to the hierarchical order.
for each job occurrence J_j^ℓ in order **do**
    Remove from the interval $(0, 2m+1]$ all subintervals assigned to "neighbors" of J_j^ℓ that precede it in the hierarchical order.
    Assign to J_j^ℓ the leftmost collection of subintervals of total size $x^\ell(j)$.
end for
Let $maxwy = 0$.
for a point y that is a right endpoint of a subinterval **do**
    Let $\phi(y) \subseteq \mathcal{L}$ be the subset of all job occurrences J_j^ℓ for which one of the subintervals in $\psi(J_j^\ell)$ contains the point y.
    Let $w(y) = \sum_{J_j^\ell \in \phi(y)} w_j$.
    if $w(y) > maxwy$ **then**
        Let $maxwy = w(y)$ and $y^* = y$.
    end if
end for
Return $\phi(y^*)$.

▶ **Theorem 2.** Flowshop_Time_Windows *yields a* $(2m+1)$*-approximation for* MaxT *on* m *machines.*

Proof. Consider $\int_0^{2m+1} w(y)dy$. By our definitions,

$$\int_0^{2m+1} w(y)dy = \sum_{j=1}^{n}\sum_{\ell=1}^{|\mathcal{L}_j|}\int_{\psi(J_j^\ell)} w_j dz_j = \sum_{j=1}^{n}\sum_{\ell=1}^{|\mathcal{L}_j|} w_j x^\ell(j) = \mathbf{w}\cdot\mathbf{x}$$

The first equality is derived by a variable substitution and the second equality follows from the second property of the mapping. Since $\int_0^{2m+1} w(y)dy = \mathbf{w}\cdot\mathbf{x}$ it follows that $(2m+1)w(y^*) \geq \mathbf{w}\cdot\mathbf{x}$. ◀

▶ **Corollary 3.** *There is a pseudo-polynomial time* $(2m+1)$*-approximation algorithm for* MaxT *on* m *machines, where* $m \geq 2$ *is some constant.*

3 Approximating MaxT on Two Machines

We now show that, with a slight degradation of the approximation ratio, we can use the algorithm presented in Section 2 to obtain a polynomial-time algorithm for $m = 2$.

We start with some notations. Consider two machines, M_1 and M_2, and each job consists of two non-preemptive segments. For notation simplicity, in the following sections, we denote the processing times of J_j on M_1 and M_2 as a_j and b_j, respectively. Recall that a job $J_j \in \mathcal{J}$ has a release time $r_j \geq 0$, a due date $d_j \geq 0$, and a weight $w_j \geq 0$. Thus, in any feasible schedule of $\mathcal{J}' \subseteq \mathcal{J}$ in the flow shop model, $J_j \in \mathcal{J}'$ is processed first for a_j time units on M_1 after its release time r_j, then processed for b_j time units on M_2 and finished no later than its due date d_j.

We distinguish between three types of jobs based on their *slackness*:

- **(i)** Small jobs $\mathcal{J}_\mathcal{S}$: Job J_j is a Small job, if it has a large slack in its time window $[r_j, d_j]$, satisfying $a_j + b_j < \frac{d_j - r_j - a_j - b_j}{n^2-1}$. Note that this implies $a_j + b_j < \frac{d_j - r_j}{n^2}$.
- **(ii)** Large jobs $\mathcal{J}_\mathcal{L}$: Job J_j is a Large job, if $a_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$ and $b_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$.
- **(iii)** Almost-Large jobs $\mathcal{J}_{\mathcal{AL}}$: Job J_j is a Almost-Large job, if it satisfies $a_j + b_j \geq \frac{d_j - r_j - a_j - b_j}{n^2-1}$ (and hence $a_j + b_j \geq \frac{d_j - r_j}{n^2}$), and also one of the following:
  - **(a)** $a_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$ and $b_j < \frac{d_j - r_j - a_j - b_j}{2n^2-2}$.
  - **(b)** $a_j < \frac{d_j - r_j - a_j - b_j}{2n^2-2}$ and $b_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$.

We modify the linear program (P) in Section 2 to solve it in polynomial time in the following steps. First, we eliminate Almost-Large jobs and replace each Almost-Large job by two Large jobs and one Small job. Then, we define the modified linear program (P_{new}) of polynomial size by identifying only a polynomial number of job occurrences for each job included in this formulation. We call these job occurrences the *selected* job occurrences. All the unselected job instances will not be scheduled (fractionally). We show that any feasible solution of (P) induces a feasible solution of (P_{new}) with a slight degradation in the value of the objective function. Finally, we show how a feasible solution of (P_{new}) can be "rounded" to a schedule whose weight is $\frac{1}{9}$ of the objective value of this feasible solution of (P_{new}). This schedule is a $(9+\epsilon)$-approximation of the optimal solution.

3.1 Eliminating the Almost-Large Jobs

We first partition the set $\mathcal{J}_{\mathcal{AL}}$ of Almost-Large jobs into two subsets $\mathcal{J}^1_{\mathcal{AL}}$ and $\mathcal{J}^2_{\mathcal{AL}}$.

- **(1)** The subset $\mathcal{J}^2_{\mathcal{AL}}$ of jobs J_j satisfying $a_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$ and $b_j < \frac{d_j - r_j - a_j - b_j}{2n^2-2}$.
- **(2)** The subset $\mathcal{J}^2_{\mathcal{AL}}$ of jobs J_j satisfying $a_j < \frac{d_j - r_j - a_j - b_j}{2n^2-2}$ and $b_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$.

Consider a job $J_j \in \mathcal{J}^1_{\mathcal{AL}}$. Let time point $t_j = d_j - a_j - (2n^2-1)b_j$, so $b_j = \frac{d_j - t_j - a_j - b_j}{2n^2-2}$. Partition the occurrences of J_j into two subsets. The first subset consists of all occurrences J^ℓ_j in which the first segment of J^ℓ_j starts (on M_1) at or after t_j, and the complement subset consists of all occurrences J^ℓ_j in which the first segment of J^ℓ_j starts processing before time t_j. We replace job J_j with three new jobs as follows.

Note that the job occurrences in the first subset are essentially the job occurrences of a new job consisting of two segments of lengths a_j, b_j with new release time t_j and due date d_j. We add such a job J_{n+j} to the input. This job is Large, since $a_j \geq \frac{d_j - t_j - a_j - b_j}{2n^2-2}$ and $b_j = \frac{d_j - t_j - a_j - b_j}{2n^2-2}$.

For the second subset of job occurrences, we ignore (for now) all the occurrences where the second segment starts before $t_j + a_j$. Note that the rest of the job occurrences in the second subset are essentially the job occurrences of two new jobs: one job consisting of a single segment of length a_j (to be processed on M_1) with release time r_j and new due date $t_j + a_j$, and a second job consisting of a single segment of length b_j (to be processed on M_2) with new release time $t_j + a_j$ and due date d_j. We add these two jobs J_{2n+j} and J_{3n+j} to the input. Since $d_j - b_j = t_j + a_j + (2n^2-2)b_j \geq t_j + a_j$ and $a_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$, we have $a_j \geq \frac{t_j + a_j - r_j - a_j}{2n^2-2}$. Thus, the job J_{2n+j} is Large. The job J_{3n+j} is Small, since $b_j = \frac{d_j - t_j - a_j - b_j}{2n^2-2} < \frac{d_j - t_j - a_j - b_j}{n^2-1}$. We make sure that J_{2n+j} and J_{3n+j} are scheduled together by modifying the linear program.

Jobs $J_j \in \mathcal{J}^2_{\mathcal{AL}}$ are handled symmetrically. Let t_j be the time point satisfying $a_j = \frac{t_j - r_j - a_j - b_j}{2n^2-2}$. Partition the occurrences of J_j into two subsets. The first subset consists of all occurrences where the second segment ends at or before t_j, and the complement subset consists of all occurrences where the second segment ends processing (on M_2) after time t_j. The job occurrences in the first subset are the same as the new job occurrences of a job with two segments of lengths a_j, b_j, release time r_j and new due date t_j. We add such a job J_{n+j} to the input. This job is Large since $a_j = \frac{t_j - r_j - a_j - b_j}{2n^2-2}$ and $b_j \geq \frac{t_j - r_j - a_j - b_j}{2n^2-2}$. For the second subset, we again ignore (for now) all the occurrences where the first segment finishes after $t_j - b_j$. Then the rest job occurrences in the second subset are the same as the job occurrences of two new jobs: one job with a single segment of length b_j (to be processed on M_2) with new release time $t_j - b_j$ and due date d_j, and a second job consisting of a single segment of length a_j (to be processed on M_1) with release time r_j and new due date $t_j - b_j$. We add these jobs J_{2n+j} and J_{3n+j} to the input. Since $t_j - b_j = r_j + a_j + (2n^2-2)a_j \geq r_j + a_j$ and $b_j \geq \frac{d_j - r_j - a_j - b_j}{2n^2-2}$, we have $b_j \geq \frac{d_j - (t_j - b_j) - b_j}{2n^2-2}$. Thus, the job J_{2n+j} is Large. Since $a_j = \frac{t_j - r_j - a_j - b_j}{2n^2-2} < \frac{(t_j - b_j) - r_j - a_j}{n^2-1}$, the job J_{3n+j} is Small.

3.2 The Selected Occurrences of Small and Large Jobs

After eliminating Almost-Large jobs, the set of Small jobs in the modified LP becomes

$$\mathcal{J}_{\mathcal{S}}{}^{new} = \mathcal{J}_{\mathcal{S}} \bigcup \{J_{3n+j} |\ J_j \in \mathcal{J}_{\mathcal{AL}}\} \text{ and } |\mathcal{J}_{\mathcal{S}}{}^{new}| = |\mathcal{J}_{\mathcal{S}}| + |\mathcal{J}_{\mathcal{AL}}|.$$

For each Small job $J_j \in \mathcal{J}_{\mathcal{S}}{}^{new}$, find n^2 non-overlapping occurrences of J_j: $J^1_j, \ldots, J^{n^2}_j$, such that in each such occurrence, the two segments of J_j are scheduled with no wait, i.e., the second segment is scheduled on M_2 immediately after completing the first segment on M_1. We can find n^2 such job occurrences since $a_j + b_j < \frac{d_j - r_j}{n^2}$. These non-overlapping occurrences are the selected occurrences of job J_j.

After eliminating Almost-Large jobs, the set of Large jobs in the modified LP is

$$\mathcal{J}_{\mathcal{L}}{}^{new} = \mathcal{J}_{\mathcal{L}} \bigcup \{J_{n+j}, J_{2n+j} |\ J_j \in \mathcal{J}_{\mathcal{AL}}\} \text{ and } |\mathcal{J}_{\mathcal{L}}{}^{new}| = |\mathcal{J}_{\mathcal{L}}| + 2|\mathcal{J}_{\mathcal{AL}}|.$$

For every $J_j \in {\mathcal{J}_{\mathcal{L}}}^{new}$, define $2n^2+1$ dividers on the time axis for machine M_1, at the time points $r_j + h \cdot \frac{(d_j - b_j - r_j)}{2n^2}$, for $h = 0, \ldots, 2n^2$, and $2n^2+1$ dividers on the time axis for machine M_2, at the time points $r_j + a_j + h \cdot \frac{(d_j - r_j - a_j)}{2n^2}$, for $h = 0, \ldots, 2n^2$. The $|{\mathcal{J}_{\mathcal{L}}}^{new}|(4n^2+2)$ dividers define half open time slots for M_1 and M_2, where each time slot is between adjacent dividers. We note that for any $J_j \in {\mathcal{J}_{\mathcal{L}}}^{new}$, no segment of J_j is completely contained in a time slot, i.e., it lies between two adjacent dividers.

Consider a LARGE job $J_j \in {\mathcal{J}_{\mathcal{L}}}^{new}$. For each time slot s for M_1 and time slot t for M_2, consider the set of all job occurrences of J_j where the right endpoint of its first segment is in time slot s and the right endpoint of its second segment is in t. Select one arbitrary job occurrence from this set. Let J_j^ℓ, for $1 \leq \ell \leq (2n^2+1)^2$, be all the selected job occurrences.

3.3 The Modified Linear Program

The set of jobs in the modified LP is $\mathcal{J}_{new} = {\mathcal{J}_{\mathcal{S}}}^{new} \bigcup {\mathcal{J}_{\mathcal{L}}}^{new}$. Thus, $|\mathcal{J}_{new}| = |\mathcal{J}_{\mathcal{S}}| + |\mathcal{J}_{\mathcal{L}}| + 3|\mathcal{J}_{\mathcal{AL}}| \leq 3n$. All the jobs in $\mathcal{J}_{new}$ are either SMALL or LARGE. We only consider variables that correspond to the selected job occurrences. We define job cliques as before but only with respect to the selected job occurrences. The modified linear program is as follows.

$$
\begin{array}{lllll}
(\mathrm{P}_{new}) \quad \text{maximize} \quad \mathbf{w} \cdot \mathbf{x} \quad \textit{subject to}: & & & & \\
\displaystyle\sum_{J_j^\ell \in \mathcal{C}} x^\ell(j) & & \leq & 2 & \text{for each clique } \mathcal{C} \\
\displaystyle\sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(j) & & \leq & 1 & \forall J_j \in \mathcal{J}_{new} \\
\displaystyle\sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(n+j) + \sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(2n+j) & & \leq & 1 & \forall J_j \in J_{\mathcal{AL}} \\
\displaystyle\sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(2n+j) - \sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(3n+j) & & = & 0 & \forall J_j \in J_{\mathcal{AL}} \\
x^\ell(j) & & \geq & 0 & \forall J_j \in \mathcal{J}_{new},\ 1 \leq \ell \leq (2n^2+1)^2
\end{array}
$$

The first constraint is a relaxation of the original clique constraint and ensures that the total value of the variables associated with the selected job occurrences in each clique is at most *two*. The second constraint guarantees that at most one occurrence of a job J_j is selected for the solution, $\forall J_j \in \mathcal{J}_{new}$. The third and fourth constraints deal with the jobs that replace the ALMOST-LARGE jobs. Recall that in Section 3.1 the occurrences of any $J_j \in \mathcal{J}_{\mathcal{AL}}$ were partitioned into two subsets. The third constraint ensures that the total value of the variables associated with the two LARGE jobs that replace a single ALMOST-LARGE job (one LARGE job for each subset of job occurrences of the ALMOST-LARGE job) is at most one. The fourth constraint ensures that for each pair of LARGE job and SMALL job that replace the second subset of job occurrences of a single ALMOST-LARGE job, the total value of the variables associated with the replacement LARGE job is the same as the total value of the variables associated with the replacement SMALL job.

3.4 The Induced Solution of the Modified Linear Program

Consider a solution of the linear program (P) for an instance with two machines. Denote this solution $y^\ell(j)$, for $1 \le j \le n$, $1 \le \ell \le \mathcal{L}_j$. We show how it induces a solution to the modified linear program (P_{new}) as follows. If $J_j \in \mathcal{J}_\mathcal{S}$, then for $\ell = 1, \ldots, n^2$, $x^\ell(j) = \frac{1}{n^2} \sum_{r=1}^{\mathcal{L}_j} y^r(j)$.

If $J_j \in \mathcal{J}_\mathcal{L}$, then for each selected job occurrence J_j^ℓ, the variable $x^\ell(j)$ is the sum of all variables $y^\ell(r)$, over all job occurrences J_j^r such that the right endpoint of the first segment of J_j^r is in the same time slot as the right endpoint of the first segment of J_j^ℓ and the right endpoint of the second segment of J_j^r is in the same time slot as the right endpoint of the second segment of J_j^ℓ.

Suppose $J_j \in \mathcal{J}^1_{\mathcal{AL}}$. Recall that in Section 3.1 the occurrences of J_j were partitioned into two subsets, denoted as S_1 and S_2. (The subset S_2 includes the job occurrences we ignored in Section 3.1.) For each selected job occurrence J_{n+j}^ℓ, the variable $x^\ell(n+j)$ is the sum of all variables $y^\ell(r)$, over all job occurrences $J_j^r \in S_1$ such that the right endpoint of the first (second) segment of J_j^r is in the same time slot as the right endpoint of the first (second) segment of J_j^ℓ. Similarly, for each selected job occurrence J_{n+2j}^ℓ, the variable $x^\ell(n+2j)$ is the sum of all variables $y^\ell(r)$, over all $J_j^r \in S_2$ such that the right endpoint of the first segment of J_j^r is in the same time slot as the right endpoint of the first segment of J_j^ℓ.

Symmetrically, suppose $J_j \in \mathcal{J}^2_{\mathcal{AL}}$. For each selected job occurrence J_{n+j}^ℓ, the variable $x^\ell(n+j)$ is defined as above. For each selected job occurrence J_{n+2j}^ℓ, the variable $x^\ell(n+2j)$ is the sum of all variables $y^\ell(r)$, over all $J_j^r \in S_2$ such that the right endpoint of the second segment of J_j^r is in the same time slot as the right endpoint of the second segment of J_j^ℓ.

Finally, suppose $J_j \in \mathcal{J}_\mathcal{AL}$ for $\ell = 1, \ldots, n^2$, we have $x^\ell(n+3j) = \frac{1}{n^2} \sum_{J_j^r \in S_2} y^r(j)$.

It is straightforward to verify that the induced solution of the modified linear program (P_{new}) satisfies all constraints but the relaxed clique constraint. We show that the relaxed clique constraint is satisfied as well. First, we show that it is satisfied when we ignore the variables associated with Small jobs.

Consider a time slot s for M_1, and assume that at least one selected job occurrence J_j^ℓ has the right endpoint of its first segment is time slot s. Let $\mathcal{C}$ be the clique defined by this endpoint. Let S be the set of all job occurrences J_k^r whose first segment intersects time slot s. Clearly, $\sum_{J_k^r \in \mathcal{C}} x^r(k) \le \sum_{J_k^r \in S} y^r(r)$. Since all jobs are Large, the first segment of any $J_k^r \in S$ intersects at least one of the dividers that define time slot s. It follows $\sum_{J_k^r \in \mathcal{C}} x^r(k) \le 2$. The same argument holds for any time slot t for M_2.

Next, we show how the relaxed clique constraint is satisfied when we add the variables associated with Small jobs. Note that for each variable associated with a Small job J_j, $x^\ell(j) \le \frac{1}{n^2}$, for $1 \le \ell \le n^2$. Still, adding these variables may render the solution infeasible. Since for each Small job J_j, the job occurrences J_j^ℓ, $1 \le \ell \le n^2$ are nonoverlapping, any job clique $\mathcal{C}$ contains at most one segment out of all segments of the job occurrences J_j^ℓ, $1 \le \ell \le n^2$. Thus, the total sum of fractions assigned to Small jobs in any job clique $\mathcal{C}$ is at most $\frac{1}{n^2} \cdot n = \frac{1}{n}$. It follows that scaling the fractions assigned to Large jobs by a factor of $(1 - \frac{1}{n})$ will make the solution feasible. This scaling degrades the value of the objective function of the fractional solution by a factor of $(1 - \frac{1}{n})$.

3.5 Rounding the Solution of the Modified Linear Program

Since the clique constraint is relaxed, we need to reformulate Lemma 1.

▶ **Lemma 4.** *Let* $\mathbf{x}$ *be a feasible solution to* (P_{new}). *Then, there exists a job occurrence* J_j^ℓ *satisfying* $x^\ell(j) + \sum_{J_k^r \in \tilde{N}(J_j^\ell)} x^r(k) \le 9$.

Proof. The proof is similar to the proof of Lemma 1. We first show that there exists a selected job occurrence J_j^ℓ for which

$$x^\ell(j) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} x^r(k) \leq 8. \tag{2}$$

As before, we define $z(J_j^\ell, J_k^r) = x^\ell(j) \cdot x^r(k)$. The analysis is slightly different from the one in the proof of Lemma 1 since the first constraint in (P_{new}) is now relaxed. We omit the details. ◀

We apply Lemma 4 to obtain a mapping (as defined in Section 2.1). This can be done in polynomial time since we are guaranteed to have a polynomial number of nonzero variables that correspond to job occurrences. The mapping yields a schedule of a subset of jobs in $\mathcal{J}_{new}$ as defined in Section 2.1 with total weight $\frac{1}{9}$ of the objective value of the feasible solution of (P_{new}). Recall that this value is the objective value of the feasible solution of (P) scaled down by a factor of $(1 - \frac{1}{n})$. We summarize in the next theorem.

▶ **Theorem 5.** *There is a polynomial time $(9+\epsilon)$-approximation algorithm for MaxT on two machines.*

4 Better Approximations for Special Cases on Two Machines

4.1 A 4-approximation Algorithm for Unit Weight Jobs with the Same Release Time

Consider instances of flow shop with two machines, in which all jobs have the same release time, i.e., $r_j = 0\ \forall J_j \in \mathcal{J}$, arbitrary due dates, and unit weight. Below, we show that for such instances a simple algorithm yields an improved approximation ratio of 4 for MaxT. We note that the problem of maximizing throughput on a *single* machine with the same release times and unit job weights is solvable in polynomial time using Moore's algorithm [19]. We call this problem below MaxT$_S$. Moore's algorithm can thus be used as a subroutine in our algorithm for MaxT, Split_the_Schedule. We give the pseudocode in Algorithm 2.

Algorithm 2 Split_the_Schedule.

1: For any job $J_j \in \mathcal{J}$ let $p_j = a_j + b_j$.
2: Solve optimally MaxT$_S$, where each job J_j has a processing time p_j, a release time $r_j = 0$, and a due date d_j. Let SOL be the set of jobs in the solution.
3: Define the following flow shop schedule of SOL on M_1 and M_2: for any $J_j \in SOL$ that is processed on the single machine in $(s_j, t_j]$, process J_j on M_1 in $(s_j, s_j + a_j]$ and on M_2 in $(s_j + a_j, t_j]$.
4: Return the schedule of SOL on M_1 and M_2.

▶ **Theorem 6.** *Let OPT be the set of jobs in an optimal solution for MaxT. Then $|SOL| \geq \frac{|OPT|}{4}$.*

We use the following two lemmas to prove our main result in Theorem 6.

▶ **Lemma 7.** *For any instance of MaxT where $r_j = 0\ \forall J_j \in \mathcal{J}$, there exists an optimal* permutation schedule, *i.e., a schedule where jobs are scheduled in the same order on both machines.*

Proof. Consider an optimal schedule that is not a permutation schedule, then we show that by swapping jobs we can obtain a feasible permutation schedule. Formally, given a schedule of the jobs on the two machines, we scan the schedule on M_2, starting from time $t = 0$. For any two consecutive jobs on M_2, J_k, J_j, if J_k precedes J_j on M_2, but J_j precedes J_k on M_1, we modify the schedule on M_1 as follows. Let s_j^1, s_k^1 the start-times of J_j, J_k on M_1, and t_k^1 the completion time of J_k on M_1 (see Figure 2). Then we schedule J_k on M_1 at time $t_k^1 - a_k - a_j$ and J_j at time $t_k^1 - a_j$. For M_1 we have:

- J_k completes processing earlier.
- J_j has a later completion time on M_1, but it still completes processing on M_2 by its due date. Indeed, as before, J_j starts processing on M_2 at time $s_j^2 \geq t_k^1$ and completes by d_j.
- For any other job J_r, $r \neq j, k$, the above swap can only result in an earlier completion time of J_r on M_1.

In addition, since we made no change on M_2, the schedule is still feasible. ◀

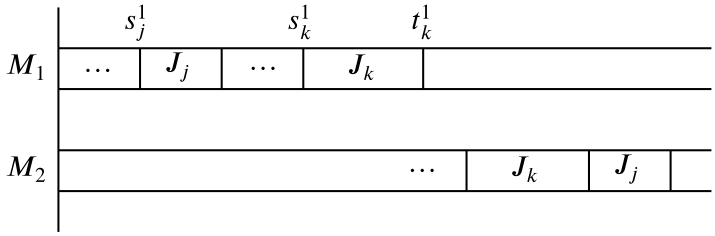


Figure 2 A non permutation schedule. J_j and J_k can be swapped on M_1 and scheduled consecutively, so that J_j completes processing on M_1 at time t_k^1.

▶ **Lemma 8.** *Let OPT be an optimal solution for a MaxT instance $\mathcal{J}$ for which there is a permutation schedule. Then there exists a subset of jobs $OPT_{single} \subseteq OPT$ satisfying:*

(i) *The jobs in OPT_{single} can be feasibly scheduled on a single machine, taking the processing time of each $J_j \in OPT_{single}$ to be $p_j = a_j + b_j$.*

(ii) $|OPT_{single}| \geq \frac{|OPT|}{4}$.

Proof. Consider an optimal subset of jobs, OPT, which has a permutation schedule. Assume w.l.o.g. that this permutation is the identity permutation. We now show how to move from a two machine schedule to a schedule of a subset of jobs in OPT, such that each job is completely processed either on M_1 or on M_2. We note that if $|OPT|$ is odd then we can always process the two segments of the last job on M_1. Hence, we assume from now on that $|OPT| = 2k$ for some integer $k \geq 1$. We now partition OPT to k pairs of jobs: $(J_1, J_2), \ldots, (J_{2i-1}, J_{2i}), \ldots$. Consider the jobs J_{2i-1}, J_{2i} with the processing times (a_{2i-1}, b_{2i-1}) and (a_{2i}, b_{2i}), respectively. We distinguish between two cases:

(i) If $a_{2i} > b_{2i-1}$ then we schedule J_{2i-1} on M_1 with processing time $p_{2i-1} = a_{2i-1} + b_{2i-1}$.

(ii) If $a_{2i} \leq b_{2i-1}$ then we schedule J_{2i} on M_2 with processing time $p_{2i} = a_{2i} + b_{2i}$.

We note that the schedules obtained on M_1 and M_2 are feasible. In addition, from each pair of jobs in OPT, one job is scheduled (either on M_1 or on M_2). Therefore, $|OPT|/2$ jobs are scheduled. Now, we choose the machine with a maximum number of jobs. This yields a solution consisting of at least $|OPT|/4$ jobs. ◀

Proof of Theorem 6. Recall that for an instance in which $r_j = 0$ $\forall J_j \in \mathcal{J}$ we can use Moore's algorithm to solve MaxTs optimally. By Lemma 8, there exists a subset of $|OPT|/4$ jobs that can be scheduled feasibly on a single machine, where OPT is an optimal solution for MaxT on two machines. Since Moore's algorithm outputs an optimal solution on a single machine, we have the statement of the theorem. ◀

4.2 A $(3+\varepsilon)$-approximation Algorithm for Jobs with the Same Release Time and Due Date

Consider instances of flow shop with two machines, in which all jobs have the same release time and the same due date. We assume below that for all $1 \leq j \leq n$, $r_j = 0$ and $d_j = T$, for some $T > 0$. We note that MaxT on such instances is NP-hard, as Knapsack is the special case where $b_j = 0$ for all $1 \leq j \leq n$.

Algorithm 3 below is a $(3+\varepsilon)$-approximation algorithm for such instances. The algorithm partitions the jobs into two groups: *large* and *small* jobs. For the large jobs the algorithm finds an optimal solution by applying an algorithm for makespan minimization in two-machine flow shop due to Johnson [16]. For the small jobs it finds a $(2+\varepsilon)$-approximation by applying a greedy algorithm for the *knapsack* problem. The algorithm then outputs the better of the two solutions, to yield a $(3+\varepsilon)$-approximation.

Let $\Delta_j = \max\{a_j, b_j\}$ for each job J_j, $1 \leq j \leq n$. Also, let $\Delta_0 = 0$. For a set of jobs $\mathcal{J}$, define the weight of $\mathcal{J}$ to be $w(\mathcal{J}) = \sum_{J_j \in \mathcal{J}} w_j$.

Algorithm 3 Pack_and_Schedule.

Fix $0 < \varepsilon < 1$.
Let $\mathcal{L} = \{J_j \in \mathcal{J} \mid \Delta_j \geq \frac{\varepsilon T}{6}\}$ and $\mathcal{S} = \mathcal{J} \setminus \mathcal{L}$.
Let $M_L = 0$
for all $R \subseteq \mathcal{L}$ such that $|R| \leq \frac{12}{\varepsilon}$ jobs **do**
  if R can be scheduled with makespan at most T **then**
    ▷ use Johnson's Algorithm [16] to check this condition
    if $w(R) > M_L$ **then**
      Let $SOL_L = R$
      Let $M_L = w(R)$
    end if
  end if
end for
Order the jobs in $\mathcal{S}$ in non-ascending order of the ratio $\frac{w_j}{\Delta_j}$. Assume w.l.o.g. that $\mathcal{S} = \{J_1, \ldots, J_{|\mathcal{S}|}\}$, and $\frac{w_1}{\Delta_1} \geq \frac{w_2}{\Delta_2} \geq \ldots \geq \frac{w_{|\mathcal{S}|}}{\Delta_{|\mathcal{S}|}}$
Find the maximum index k such that $\sum_{j=1}^{k} \Delta_j \leq T(1 - \frac{\varepsilon}{6})$
Let $SOL_S = \{J_1, \ldots, J_k\}$ ▷ SOL_S can be scheduled feasibly as shown below
If $w(SOL_L) > w(SOL_S)$ then SOL $= SOL_L$; else SOL $= SOL_S$.
Return SOL

▶ **Theorem 9.** *For any fixed $0 < \varepsilon < 1$, Algorithm 3 runs in polynomial time and yields a $(3+\varepsilon)$-approximation for MaxT on instances where $r_j = 0$ and $d_j = T$, $\forall j$.*

To prove Theorem 9 we need an observation and a few lemmas.

▶ **Observation 10.** Any feasible solution R of MaxT on instances where $r_j = 0$ and $d_j = T$ $\forall j$ satisfies $\sum_{J_j \in R} \Delta_j \leq 2T$.

Proof. We note that $\sum_{J_j \in R} \Delta_j = \sum_{J_j \in R} \max\{a_j, b_j\} \leq \sum_{J_j \in R} (a_j + b_j) \leq 2T$. The last inequality follows from the fact that R can be scheduled feasibly. ◀

▶ **Lemma 11.** *The set SOL_L is an optimal solution for input $\mathcal{L}$.*

Proof. Since for every job $J_j \in \mathcal{L}$ we have $\Delta_j \geq \frac{\varepsilon T}{6}$, it follows from Observation 10 that any feasible solution for $\mathcal{L}$ cannot include more than $\frac{12}{\varepsilon}$ jobs. Since we enumerate over all feasible schedules with up to this number we are guaranteed to find the optimum. ◀

▶ **Lemma 12.** *The jobs in $SOL_S = \{J_1, \ldots, J_k\}$ can be scheduled feasibly.*

Proof. Sort the jobs $J_1, \ldots, J_k$ in non-ascending order of Δ_j. Let π be the resulting permutation; that is, $\Delta_{\pi(1)} \geq \Delta_{\pi(2)} \geq \ldots \geq \Delta_{\pi(k)}$. Let $\pi(0) = 0$.

Schedule job $J_{\pi(j)}$ at time $t_1^j = \sum_{i=0}^{j-1} \Delta_{\pi(i)}$ on M_1 and at time $t_2^j = t_1^j + \Delta_{\pi(1)}$ on M_2. The schedule is feasible since (i) no two jobs overlap in any of the machines (recall that $\Delta_j = \max\{a_j, b_j\}$), (ii) the makespan of the schedule is $\Delta_{\pi(1)} + \sum_{j=1}^{k} \Delta_{\pi(j)} \leq T$ since $\Delta_{\pi(1)} < \frac{\varepsilon T}{6}$, and (iii) for any job J_j, its segment on M_2 is executed after the completion of its segment on M_1, since the schedule on M_2 is shifted by $\Delta_{\pi(1)} = \max_{j \in [1..k]} \Delta_j$. ◀

▶ **Lemma 13.** *The set SOL_S is a $(2+\varepsilon)$-approximation of the optimal solution for input $\mathcal{S}$.*

Proof. First, consider a knapsack problem with set of items corresponding to the jobs in $\mathcal{S}$, where the size of item j is Δ_j and its weight is w_j. Assume that the knapsack capacity is T. We claim that the weight of the optimal solution to this knapsack problem has weight that is at least $\frac{1}{2}$ of the weight of optimal solution for input $\mathcal{S}$. To see this consider an optimal solution for input $\mathcal{S}$ and partition the set of jobs in this solution into two disjoint sets: the first set O_1 consists of all jobs J_j in the solution for which $\Delta_j = a_j$, and the second set O_2 consists of all jobs J_j in the solution for which $\Delta_j > a_j$ (and $\Delta_j = b_j$). Let i be the index of the set whose total weight is larger; that is $w(O_i) \geq w(O_{3-i})$. Clearly $w(O_i)$ is at least half the optimum. Since we start from a feasible solution, the total size of the items corresponding to the jobs in O_i is bounded by T. Thus, there is a feasible solution to the knapsack problem with weight that is at least $\frac{1}{2}$ of the weight of optimal solution for input $\mathcal{S}$. Note that we may not be able to feasibly schedule the set of jobs corresponding to the items in an optimal solution of this knapsack problem.

From the way we chose k and since for all $J_j \in \mathcal{S}$, $\Delta_j < \frac{\varepsilon T}{6}$, it follows that $\sum_{j=1}^{k} \Delta_j > T(1 - \frac{\varepsilon}{3})$. Since the jobs are sorted in non-ascending order of the ratio $\frac{w_j}{\Delta_j}$, we are guaranteed that $w(SOL_S)$ is at least $(1 - \frac{\varepsilon}{3})$ of the weight of the optimal solution to the knapsack problem and thus it is at least $\frac{1}{2}(1 - \frac{\varepsilon}{3})$ of the weight of the optimal solution for input $\mathcal{S}$. Since $(2+\varepsilon) \cdot \frac{1}{2}(1 - \frac{\varepsilon}{3}) = 1 + \frac{1}{6}(\varepsilon - \varepsilon^2) \geq 1$, for $0 < \varepsilon < 1$, it follows that SOL_S is a $(2+\varepsilon)$-approximation of the optimal solution for input $\mathcal{S}$. ◀

▶ **Lemma 14.** *The time complexity of Algorithm 3 is $O(n^{\frac{12}{\varepsilon}})$.*

Proof of Theorem 9. Consider an optimal solution O for input $\mathcal{J}$, and partition the jobs in this solution into two disjoint sets $O_L = O \cap \mathcal{L}$ and $O_S = O \cap \mathcal{S}$. By Lemmas 11 and 13, we have that $w(O_L) \leq w(\mathsf{SOL})$ and $w(O_S) \leq (2+\varepsilon) \cdot w(\mathsf{SOL})$. It follows that $w(O) \leq (3+\varepsilon) \cdot w(\mathsf{SOL})$.

By the algorithm and Lemma 12 the jobs in SOL can be scheduled feasibly, and by Lemma 14 the running time is polynomial in n. The theorem follows. ◀

▶ **Corollary 15.** *If $a_j \leq b_j$, for all $J_j \in \mathcal{J}$, or $a_j \geq b_j$, for all $J_j \in \mathcal{J}$, then, for any fixed $0 < \varepsilon < 1$, Algorithm 3 is a $(2+\varepsilon)$-approximation algorithm for MaxT on instances where $r_j = 0$ and $d_j = T$, $\forall j$.*

Proof. It is easy to see that if any of the conditions in the corollary hold then the weight of an optimal solution to the knapsack problem defined in Lemma 13 is at least the weight of the optimal solution for input $\mathcal{S}$, and thus the set SOL_S is a $(1+\varepsilon)$-approximation of the optimal solution for input $\mathcal{S}$. ◀

References

1 Kunal Agrawal, Jing Li, Kefu Lu, and Benjamin Moseley. Scheduling parallelizable jobs online to maximize throughput. In *Latin American Symposium on Theoretical Informatics*, pages 755–776. Springer, 2018.

2 Nikhil Bansal, Ho-Leung Chan, Rohit Khandekar, Kirk Pruhs, Baruch Schieber, and Cliff Stein. Non-preemptive min-sum scheduling with resource augmentation. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 614–624. IEEE, 2007.

3 Amotz Bar-Noy, Reuven Bar-Yehuda, Ari Freund, Joseph Naor, and Baruch Schieber. A unified approach to approximating resource allocation and scheduling. *J. ACM*, 48(5):1069–1090, 2001.

4 Amotz Bar-Noy, Sudipto Guha, Joseph Naor, and Baruch Schieber. Approximating the throughput of multiple machines in real-time scheduling. *SIAM J. Comput.*, 31(2):331–352, 2001.

5 Reuven Bar-Yehuda, Magnús M Halldórsson, Joseph Naor, Hadas Shachnai, and Irina Shapira. Scheduling split intervals. *SIAM Journal on Computing*, 36(1):1–15, 2006.

6 Michael G Bechtel, Elise McEllhiney, Minje Kim, and Heechul Yun. Deeppicar: A low-cost deep neural network-based autonomous car. In *RTCSA*, pages 11–21, 2018.

7 Bo Chen, Chris N Potts, and Gerhard J Woeginger. A review of machine scheduling: Complexity, algorithms and approximability. In *Handbook of combinatorial optimization*, pages 1493–1641. Springer, 1998.

8 Byung-Cheon Choi and Suk-Hun Yoon. Maximizing the weighted number of just-in-time jobs in flow shop scheduling. *Journal of Scheduling*, 10(4-5):237–243, 2007.

9 Julia Chuzhoy, Rafail Ostrovsky, and Yuval Rabani. Approximation algorithms for the job interval selection problem and related scheduling problems. *Mathematics of Operations Research*, 31(4):730–738, 2006.

10 Amir Elalouf, Eugene Levner, and Huajun Tang. An improved FPTAS for maximizing the weighted number of just-in-time jobs in a two-machine flow shop problem. *Journal of Scheduling*, 16(4):429–435, 2013.

11 Michael R Garey and David S Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

12 Michael R Garey, David S Johnson, and Ravi Sethi. The complexity of flowshop and jobshop scheduling. *Mathematics of operations research*, 1(2):117–129, 1976.

13 Leslie A. Hall. Approximability of flow shop scheduling. *Math. Program.*, 82:175–190, 1998.

14 Elad Hazan, Shmuel Safra, and Oded Schwartz. On the complexity of approximating k-dimensional matching. In *Proceedings of APPROX*, pages 83–97. Springer, 2003.

15 Sungjin Im, Shi Li, Benjamin Moseley, and Eric Torng. A dynamic programming framework for non-preemptive scheduling problems on multiple machines. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 1070–1086. SIAM, 2014.

16 Selmer Martin Johnson. Optimal two-and three-stage production schedules with setup times included. *Naval research logistics quarterly*, 1(1):61–68, 1954.

17 Bala Kalyanasundaram and Kirk Pruhs. Speed is as powerful as clairvoyance. *Journal of the ACM (JACM)*, 47(4):617–643, 2000.

18 Monaldo Mastrolilli and Ola Svensson. Hardness of approximating flow and job shop scheduling problems. *Journal of the ACM (JACM)*, 58(5):20, 2011.

19 J Michael Moore. An n job, one machine sequencing algorithm for minimizing the number of late jobs. *Management science*, 15(1):102–109, 1968.

20 Viswanath Nagarajan and Maxim Sviridenko. Tight bounds for permutation flow shop scheduling. *Mathematics of Operations Research*, 34(2):417–427, 2009.

21 Sartaj K Sahni. Algorithms for scheduling independent tasks. *Journal of the ACM (JACM)*, 23(1):116–127, 1976.

22 Jeanette P. Schmidt, Alan Siegel, and Aravind Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995.

23 Dvir Shabtay and Yaron Bensoussan. Maximizing the weighted number of just-in-time jobs in several two-machine scheduling systems. *Journal of Scheduling*, 15(1):39–47, 2012.

24 Maxim Sviridenko. A note on permutation flow shop problem. *Annals of Operations Research*, 129(1-4):247–252, 2004.

25 Gerhard J Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.

26 Yiyong Xiao, Siyue Zhang, Pei Yang, Meng You, and Jiaoying Huang. A two-stage flow-shop scheme for the multi-satellite observation and data-downlink scheduling problem considering weather uncertainties. *Reliability Engineering & System Safety*, 188:263–275, 2019.

A Some Proofs

Proof of Lemma 1. We first show that there exists a job occurrence J_j^ℓ for which the following holds:

$$x^\ell(j) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} x^r(k) \leq 2m. \tag{3}$$

For two "neighboring" job occurrences J_j^ℓ and J_k^r (i.e., $J_k^r \in \tilde{N}_1(J_j^\ell)$ and $J_j^\ell \in \tilde{N}_1(J_k^r)$), define $z(J_j^\ell, J_k^r) = x^\ell(j) \cdot x^r(k)$. We also define $z(J_j^\ell, J_j^\ell) = (x^\ell(j))^2$. We prove (3) using a weighted averaging argument, where the weights are the values $z(J_j^\ell, J_k^r)$ for all pairs of job occurrences which have intersecting segments.

Consider the sum $\sum_{j=1}^{n} \sum_{\ell=1}^{\mathcal{L}_j} \left(z(J_j^\ell, J_j^\ell) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} z(J_j^\ell, J_k^r) \right)$. We upper bound this sum as follows. Let $\mathcal{I}(J_j^\ell)$ denote the set of segments of a job occurrence J_j^ℓ. For each job occurrence J_j^ℓ, we consider all of its segments $I \in \mathcal{I}(J_j^\ell)$. For each such segment I, we sum up $z(J_j^\ell, J_k^r)$ for all job occurrences J_k^r having at least one segment that intersects with I (including J_j^ℓ itself). Let $R(J_j^\ell, I)$ be the set of job occurrences that have a segment intersecting the right endpoint of I (including J_j^ℓ itself). We note that it suffices to sum up $z(J_j^\ell, J_k^r)$ only for job occurrences $J_k^r \in R(J_j^\ell, I)$ and then multiply the total sum by 2. This is because, for the intersecting segment I of J_j^ℓ and segment I' of J_k^r, if the right endpoint of I precedes the right endpoint of I', then $J_k^r \in R(J_j^\ell, I)$; otherwise, $J_j^\ell \in R(J_k^r, I')$. Since $z(J_j^\ell, J_k^r) = z(J_k^r, J_j^\ell)$, each of them contributes the same value to the other. Therefore, it follows that

$$\sum_{j=1}^{n} \sum_{\ell=1}^{|\mathcal{L}_j|} \left(z(J_j^\ell, J_j^\ell) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} z(J_j^\ell, J_k^r) \right) \leq 2 \sum_{j=1}^{n} \sum_{\ell=1}^{|\mathcal{L}_j|} \sum_{I \in \mathcal{I}(J_j^\ell)} \sum_{J_k^r \in R(J_j^\ell, I)} z(J_j^\ell, J_k^r). \tag{4}$$

By the first constraint in (P), the definition of job cliques and the definition of $z(J_j^\ell, J_k^r)$, we have

$$\sum_{J_k^r \in R(J_j^\ell, I)} z(J_j^\ell, J_k^r) \leq \; x^\ell(j) \cdot \sum_{J_k^r \in R(J_j^\ell, I)} x^r(k) \leq x^\ell(j). \tag{5}$$

Using (4), (5), and the fact that $|\mathcal{I}(J_j^\ell)| \leq m$, we get that

$$\sum_{j=1}^{n} \sum_{\ell=1}^{|\mathcal{L}_j|} \left(z(J_j^\ell, J_j^\ell) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} z(J_j^\ell, J_k^r) \right) \leq 2 \sum_{j=1}^{n} \sum_{\ell=1}^{|\mathcal{L}_j|} \sum_{I \in \mathcal{I}(J_j^\ell)} x^\ell(j) \leq 2m \sum_{j=1}^{n} \sum_{\ell=1}^{|\mathcal{L}_j|} x^\ell(j).$$

Hence, there exists a job occurrence J_j^ℓ satisfying

$$z(J_j^\ell, J_j^\ell) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} z(J_j^\ell, J_k^r) = (x^\ell(j))^2 + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} x^r(k)x^\ell(j) \leq 2m \cdot x^\ell(j).$$

By factoring out $x^\ell(j)$ from both sides we get inequality (3).

To complete the proof of the lemma, we note that for a job J_j^ℓ satisfying (3) it also holds that

$$x^\ell(j) + \sum_{J_k^r \in \tilde{N}(J_j^\ell)} x^r(k) = x^\ell(j) + \sum_{J_k^r \in \tilde{N}_1(J_j^\ell)} x^r(k) + \sum_{J_k^r \in \tilde{N}_2(J_j^\ell)} x^r(k) \leq 2m + \sum_{r=1}^{|\mathcal{L}_j|} x^r(j) \leq 2m+1.$$

The last inequality follows from the second constraint in (P). ◀

Proof of Lemma 14. It is easy to see that the most time consuming part is the loop defined in Step 4 where we enumerate over all subsets of $\mathcal{L}$ of size at most $\frac{12}{\varepsilon}$ and thus the time complexity. ◀

Maximizing the Correlation: Extending Grothendieck's Inequality to Large Domains

Dor Katzelnick
Department of Computer Science, Technion, Haifa, Israel
dkatzelnick@cs.technion.ac.il

Roy Schwartz
Department of Computer Science, Technion, Haifa, Israel
schwartz@cs.technion.ac.il

Abstract

Correlation Clustering is an elegant model where given a graph with edges labeled + or −, the goal is to produce a clustering that agrees the most with the labels: + edges should reside within clusters and − edges should cross between clusters. In this work we study the MaxCorr objective, aiming to find a clustering that maximizes the difference between edges classified correctly and incorrectly. We focus on the case of bipartite graphs and present an improved approximation of 0.254, improving upon the known approximation of 0.219 given by Charikar and Wirth [FOCS'2004] and going beyond the 0.2296 barrier imposed by their technique. Our algorithm is inspired by Krivine's method for bounding Grothendieck's constant, and we extend this method to allow for more than two clusters in the output. Moreover, our algorithm leads to two additional results: (1) the first known approximation guarantees for MaxCorr where the output is constrained to have a bounded number of clusters; and (2) a natural extension of Grothendieck's inequality to large domains.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Mathematics of computing → Combinatorial optimization

Keywords and phrases Correlation Clustering, Grothendieck's Inequality, Approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.49

Category APPROX

Funding *Roy Schwartz*: Research is supported by ISF grant 1336/16.

1 Introduction

Correlation Clustering is a classic model, where given objects and possibly inconsistent pairwise similarity (or dissimilarity) information regarding the objects, the goal is to cluster the objects in a way that agrees the most with the given information. Since its introduction by Bansal et al. [7] close to two decades ago, Correlation Clustering has found numerous practical applications in a wide range of settings: image segmentation [35], cross-lingual link detection [33], clustering gene expression patterns [5, 8], coreference resolution [14, 15, 28] and aggregating inconsistent clusterings [18], to name a few (refer to the survey [35] and the references therein for additional details). Moreover, from a theoretical perspective it can be shown that Correlation Clustering captures classic graph cuts problems, including Min $s-t$ Cut, Multiway Cut and Multicut.

Formally, in Correlation Clustering we are given an undirected graph $G = (V, E)$ equipped with non-negative edge weights $w : E \to \mathbb{R}_+$. Additionally, each edge is labeled either with a + or a −, where + indicates similarity and − indicates dissimilarity. We consider E to be the disjoint union of E^+ (all edges labeled +) and E^- (all edges labeled −). Intuitively, two nodes connected by a + edge (− edge) are said to be similar (dissimilar) and the weight of the edge quantifies the strength of the similarity (dissimilarity) between the


© Dor Katzelnick and Roy Schwartz;
licensed under Creative Commons License CC-BY



two nodes. A *clustering* of the graph is a partition $\mathcal{C} = \{S_1, \ldots, S_\ell\}$ of V, for some ℓ, where each S_i is called a cluster. Two nodes u and v are in *agreement* with respect to a clustering $\mathcal{C}$ if $(u,v) \in E^+$ (or $(u,v) \in E^-$) and u and v belong to the same cluster (different clusters), and are in *disagreement* otherwise.

The above leads to three natural objectives, which were introduced in the original paper of Bansal *et. al.* [7], measuring the compliance of a clustering $\mathcal{C}$ with the input: MaxAgree, MinDisagree and MaxCorr. Given a clustering $\mathcal{C}$, the first objective aims to maximize the total weight of edges that are in agreement, the second objective aims to minimize the total weight of edges that are in disagreement, and the third objective combines the previous two objectives and aims to maximize the difference between the total weight of edges in agreement and disagreement. Formally, given a clustering $\mathcal{C}$ the MaxCorr objective value of $\mathcal{C}$ is denoted by $\text{Corr}(\mathcal{C})$ and is defined as follows: $\text{Corr}(\mathcal{C}) \triangleq \text{Agree}(\mathcal{C}) - \text{DisAgree}(\mathcal{C})$, where:

$$\text{Agree}(\mathcal{C}) = \sum_{(u,v)\in E^+:\mathcal{C}(u)=\mathcal{C}(v)} w_{u,v} + \sum_{(u,v)\in E^-:\mathcal{C}(u)\neq\mathcal{C}(v)} w_{u,v}$$

$$\text{DisAgree}(\mathcal{C}) = \sum_{(u,v)\in E^+:\mathcal{C}(u)\neq\mathcal{C}(v)} w_{u,v} + \sum_{(u,v)\in E^-:\mathcal{C}(u)=\mathcal{C}(v)} w_{u,v}.$$

In the above $\mathcal{C}(u)$ denotes the cluster in $\mathcal{C}$ that u belongs to. The goal is to find a clustering $\mathcal{C}$ that maximizes $\text{Corr}(\mathcal{C})$.[1] A generalization of MaxCorr, denoted by Max-k-Corr, is where we are also given as input a parameter k that upper bounds the possible number of clusters in $\mathcal{C}$.

All the above objectives have been extensively studied for almost two decades for important special cases, such as bipartite graphs [1, 5, 6, 12], complete unweighted graphs [7, 10] and weights satisfying specific constraints [2, 12], as well as general graphs [10, 11, 17, 32]. The more general problem where the number of clusters is bounded has also been studied, e.g., [7, 16, 20, 24, 6]. From a practical perspective, the study of clustering of bipartite graphs was motivated by numerous applications, such as gene expression and biological data analysis [13, 27], and data mining applications [36].

The only known algorithm for MaxCorr is given by Charikar and Wirth [11] who elegantly reduced the problem to maximizing a quadratic form. We denote the latter by Max Quad: given a matrix $B \in \mathbb{R}^{n\times n}$ find $\mathbf{x} \in \{\pm 1\}^n$ maximizing $\mathbf{x}^T B \mathbf{x}$.[2] The algorithm of [11] works as follows (for simplicity of presentation we assume that $V = \{1, \ldots, n\}$): First, set $B_{i,j}$ to be $w_{i,j}$ if $(i,j) \in E^+$, $-w_{i,j}$ if $(i,j) \in E^-$ (and 0 otherwise) and approximately solve $\max_{\mathbf{x}\in\{\pm1\}^n}\{\mathbf{x}^T B\mathbf{x}\}$. Second, consider the clustering $\mathcal{C}_1 = \{S_1, S_2\}$ defined by $\mathbf{x}$: $S_1 = \{i : x_i = 1\}$ and $S_2 = \{i : x_i = -1\}$. Third, consider the clustering $\mathcal{C}_2 = \{S_1, \ldots, S_n\}$ of V into singletons, where $S_i = \{i\}$. Finally, return the best from $\mathcal{C}_1$ and $\mathcal{C}_2$, i.e., $\max\{\text{Corr}(\mathcal{C}_1), \text{Corr}(\mathcal{C}_2)\}$. [11] proved that if there exists an α approximation algorithm for Max Quad, the above reduction provides an approximation of $\alpha/(2+\alpha)$ for MaxCorr. Since Max Quad is known to have a logarithmic approximation [11, 31], i.e., $\alpha = \Omega(1/\log n)$, this results in an approximation of $\Omega(1/\log n)$ for MaxCorr on general graphs.

We note that using the above exact same reduction, one can implicitly deduce an improved approximation algorithm for MaxCorr for bipartite graphs by simply substituting the approximation algorithm for Max Quad with one for maximizing a bipartite quadratic form.

---

[1] Equivalently one can formulate MaxCorr without the edge labels by negating the weight of $-$ edges and then maximizing the total weight of edges inside clusters minus the total weight of edges that cross between clusters.

[2] It is assumed that the diagonal of B is all zeros, i.e., $B_{i,i} = 0$ for every $i = 1, \ldots, n$.

We denote the latter by Max BiQuad: given $A \in \mathbb{R}^{n\times m}$ find $\mathbf{x} \in \{\pm 1\}^n$ and $\mathbf{y} \in \{\pm 1\}^m$ maximizing $\mathbf{x}^T A\mathbf{y}$.[3] Specifically, assuming G is a bipartite graph containing n vertices on one side and m vertices on the other, instead of the previously defined B consider a matrix $A \in \mathbb{R}^{n\times m}$ as follows: $A_{i,j}$ is set to $w_{i,j}$ if $(i,j) \in E^+$, $-w_{i,j}$ if $(i,j) \in E^-$ and 0 otherwise. Given A, approximately solve the instance of Max BiQuad defined by A, i.e., $\max_{\mathbf{x}\in\{\pm1\}^n,\mathbf{y}\in\{\pm1\}^m}\{\mathbf{x}^T Ay\}$. The rest of [11]'s reduction remains unchanged. One can easily verify that the overall approximation guarantee of the reduction remains as before. Max BiQuad is known to have an approximation of ≈ 0.5611 [4, 26] that follows from Grothendieck's inequality, resulting in an improved approximation of $0.5611/(2+0.5611) = 0.219$ for MaxCorr when G is a bipartite graph.[4]

To the best of our knowledge, no other approximation algorithm for MaxCorr besides Charikar and Wirth [11] is known, both for general and bipartite graphs. It is important to note that when restricting attention to bipartite graphs, the algorithm of [11] imposes an intrinsic barrier of 0.2296 that follows from lower bounds on Grothendieck's constant K_G (see Section 1.2 for more details). Thus, it seems there is no much room to improve the current 0.219 approximation using [11]'s approach. Moreover, no approximation is known for Max-k-Corr, even when considering bipartite graphs. The reason for the latter is that the algorithm of [11] might output n singleton clusters, thus violating the bound k on the number of clusters. The only exception is Max-2-Corr, which coincides with Max Quad and Max BiQuad for general and bipartite graphs, respectively.

Grothendieck's Inequality with Large Domains

Our work closely relates to Grothendieck's inequality. This classic inequality, first presented in [21], states that there is a universal constant K_G such that for every matrix $A \in \mathbb{R}^{n\times m}$ (recall that S^d denotes the Euclidean unit sphere in $\mathbb{R}^{d+1}$):

$$\max_{\{\mathbf{u}_i\}_{i=1}^n \cup \{\mathbf{v}_j\}_{j=1}^m \subseteq S^{n+m-1}} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} \langle \mathbf{u}_i, \mathbf{v}_j \rangle \right\} \leq K_G \cdot \max_{\mathbf{x}\in\{\pm1\}^n,\mathbf{y}\in\{\pm1\}^m} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} x_i y_j \right\}.$$

The problem of bounding K_G, both upper and lower bounds, has been studied for more than half a century [9, 21, 26, 29, 30]. Specifically, upper bounding K_G is typically achieved by providing an approximation algorithm for Max BiQuad that rounds a fractional solution to the natural semi-definite relaxation: each x_i and y_j is assigned a unit vector $\mathbf{u}_i$ and $\mathbf{v}_j$ respectively and $x_i y_j$ is replaced with $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$. Intuitively, K_G^{-1} can be viewed as the integrality gap of this relaxation.

One can easily note that the existence of a $1/\beta_k$ approximation for Max-k-Corr for bipartite graphs, that is based on rounding the natural semi-definite relaxation (see Section 2), implies a generalization of Grothendieck's inequality to larger domains. This generalization states that there exists an α_k, where $\alpha_k \leq \beta_k$, such that for every matrix $A \in \mathbb{R}^{n\times m}$,

$$\max_{\{\mathbf{u}_i\}_{i=1}^n \cup \{\mathbf{v}_j\}_{j=1}^m \in \mathcal{P}_k} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} \frac{(2(k-1)\langle \mathbf{u}_i, \mathbf{v}_j \rangle - (k-2))}{k} \right\} \leq$$
$$\alpha_k \cdot \max_{\mathbf{x}\in[k]^n,\mathbf{y}\in[k]^m} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} s(x_i, y_j) \right\}.$$

[3] Both problems, Max BiQuad and Max Quad, are known to be NP-hard [4, 11].

[4] Braverman et al. [9] showed an improvement over 0.5611 by a very small absolute constant, thus one can in fact obtain an approximation for MaxCorr in bipartite graphs that is slightly better than 0.219.

In the above $\mathcal{P}_k$ is defined as follows: $\mathcal{P}_k \triangleq \{\mathcal{U} \subseteq S^{n+m-1} : \langle \mathbf{u}, \mathbf{v} \rangle \geq -1/(k-1)\ \forall \mathbf{u}, \mathbf{v} \in \mathcal{U}\}$, s is the signed indicator function, i.e., $s(x_i, y_j) = 1$ if $x_i = y_j$ and $s(x_i, y_j) = -1$ if $x_i \neq y_j$, and $[k] = \{0, \ldots, k-1\}$. It is important to note that when $k = 2$ the above inequality reduces to the classic Grothendieck's inequality since $\mathcal{P}_2$ only enforces that all vectors $\{\mathbf{u}_i\}_{i=1}^n$ and $\{\mathbf{v}_j\}_{j=1}^m$ are unit vectors. Thus, Grothendieck's original inequality corresponds to the case where the size of the domain is 2. Hence, for larger values of k the above inequality can be viewed as an extension of Grothendieck's original inequality to larger domains. To the best of our knowledge, such an extension was not considered in the past. A special case of particular interest of the above extended inequality, in addition to the case $k = 2$, is when $k \to \infty$. In this case the above extended inequality reduces to:

$$\max_{\{\mathbf{u}_i\}_{i=1}^n \cup \{\mathbf{v}_j\}_{j=1}^m \in \mathcal{P}_\infty} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} \left(2 \langle \mathbf{u}_i, \mathbf{v}_j \rangle - 1\right) \right\} \leq \alpha_\infty \cdot \max_{\mathbf{x} \in [n+m]^n, \mathbf{y} \in [n+m]^m} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} s(x_i, y_j) \right\},$$

where $\mathcal{P}_\infty$ only enforces that all vectors are unit and in the positive orthant. This case is interesting since one can easily observe that if MaxCorr admits an approximation of $1/\beta$ for bipartite graphs, that is based on rounding the natural semi-definite relaxation (see Section 2), then $\alpha_\infty \leq \beta$.

CSP Over Large Domains

We note that MaxCorr and Max-k-Corr can be equivalently cast as a constraint satisfaction problem over a large domain with negative payoffs. Specifically, when considering bipartite graphs, we are given variables $x_1, \ldots, x_n, y_1, \ldots, y_m$, and weighted equality/inequality constraints containing exactly two variables per constraint (one from each type), i.e., $x_i = y_j$ or $x_i \neq y_j$. Given an assignment of values from the domain $[k]$ to each of the variables, the value of a constraint is its weight if it is satisfied and the negation of its weight if it is unsatisfied. The goal is to find an assignment maximizing the sum of values of the constraints. The above equivalence obviously applies to general graphs as well. CSP over large (non-binary) domains was studied, for example, by Guruswami and Raghavendra [22] (see the references therein for additional related work). Moreover, the classic problem of Max k-Cut, studied by Frieze and Jerum [19], is also a CSP (or equivalently a clustering problem) over a domain of size k.

1.1 Our Results

In this work we focus on MaxCorr and Max-k-Corr in bipartite graphs. We obtain the following main result for MaxCorr.

▶ **Theorem 1.** *There exists a polynomial-time* 0.254*-approximation algorithm for the problem of* MaxCorr *on bipartite graphs.*

There are three important things to note regarding Theorem 1. First, it improves upon the previously known approximation of 0.219 of Charikar and Wirth [11]. Second, it goes beyond the 0.2296 barrier which is intrinsic to the algorithm of [11] and follows from lower bounds on Grothendieck's constant K_G (see Section 1.2 for more details). Third, our algorithm that proves Theorem 1 produces at most three clusters, whereas the optimal solution might have any number of clusters.

We show how our main algorithm can be adapted to handle any bound k on the number of clusters. This results in an approximation factor that for $k = 2$ equals Krivine's [4, 26] guarantee of 0.5611, gracefully degrades as k increases, and reaches the guarantee of 0.254 of Theorem 1 for unbounded k (see Table 1 for approximation guarantees for some values of k). To the best of our knowledge, no previous approximation for Max-k-Corr for bipartite graphs is known. Similarly to Theorem 1, the number of clusters produced by our algorithm for Max-k-Corr in bipartite graphs does not exceed four, regardless of how many clusters are in the optimal solution. Moreover, allowing our algorithms to produce more clusters than four is not beneficial.

Table 1 Approximation for Max-k-Corr.

k	2	3	4	5	6	10
approximation	0.561	0.397	0.348	0.32	0.309	0.285

The above approximation guarantees of MaxCorr and Max-k-Corr for bipartite graphs lead to the following results regarding α_∞ and α_k for the extension of Grothendieck's inequality to large domains.

▶ **Theorem 2.** *α_k is a universal constant for every $k \geq 2$. Moreover, α_∞ is also a universal constant.*

The specific values of the bounds on α_∞ and α_k can be derived directly from our approximation factors and integrality gaps.

1.2 Our Techniques

The existing approach for approximating MaxCorr in bipartite graphs, i.e., the reduction of Charikar and Wirth [11], poses two difficulties. The first main difficulty is that this approach most likely cannot provide a much better approximation than its promised 0.219 guarantee. Recall that its approximation equals $\alpha/(2+\alpha)$, where α is the known approximation for Max BiQuad. It can be shown that the dependence on α in the approximation is tight, i.e., [11]'s algorithm cannot provide an approximation better than $\alpha/(2+\alpha)$. Thus, improving this algorithm requires improving α, a task that most likely will improve Grothendieck's constant K_G as well. Moreover, the known lower bound of $\beta = 1.6769$ on K_G [29] yields that [11]'s approach cannot provide an approximation better than $(1/\beta)/(2+1/\beta) \approx 0.2296$ for MaxCorr on bipartite graphs. The latter is true since the only known method for approximating Max BiQuad uses the standard semi-definite relaxation whose integrality gap is exactly K_G^{-1}. We note that our guarantee of 0.254 in Theorem 1 goes beyond this 0.2296 barrier that is inherent to the approach of [11]. The second main difficulty is that the approach of Charikar and Wirth [11] results in an algorithm that might produce up to n clusters. Therefore, it cannot be applied to Max-k-Corr as it violates the constraint on the number of clusters k in the output.

To eschew the above difficulties, we adopt an approach that is inspired by Krivine's result for bounding K_G [26]. This approach is based on transforming two collections of disjoint vectors in such a way that allows random hyperplane rounding to be successful even when negative values are present in the objective function. We first show that a straightforward adaptation of this method yields an approximation of 0.198 for MaxCorr on bipartite graphs. To obtain our main result for MaxCorr we show that it suffices to extend Krivine's method to only three clusters and present suitable transformations to this end. Focusing on Max-k-Corr we require more subtlety, as we build upon and adapt our two algorithms for

MaxCorr which are based on Krivine's method (the straightforward adaptation as well as the three cluster extension). We note that allowing our algorithms to produce more than four clusters is not beneficial.

1.3 Related Work

Many variants of Correlation Clustering were presented and studied throughout the years, both from theoretical and practical perspectives. We present here only some of the more relevant previous work. For general graphs, [7, 32] presented a 0.766-approximation for MaxAgree, [10, 17] presented an $O(\log n)$ approximation for MinDisagree, [11] presented an $\Omega(1/\log n)$ approximation for MaxCorr (while [3] presented a more refined guarantee of $\Omega(1/\log(\vartheta(\bar{G})))$).

For complete unweighted graphs, Bansal *et al.* [7] presented a PTAS for MaxAgree, a constant factor approximation for MinDisagree, and a simple 3-approximation for MinDisagree restricted to two clusters. The approximation factor for MinDisagree was continually improved by Charikar et al. [10] to 4, Ailon et al. [2] to 2.5 and by Chawla et al. [12] to $2.06 - \varepsilon$. Moreover, Giotis et al. [20] presented a PTAS for MaxAgree and MinDisagree when the number of clusters is restricted to a constant k.

Focusing on bipartite graphs, Asteris et al. [6] presented a PTAS for MaxAgree on complete bipartite graphs, even when the number of clusters is restricted. Amit [5] presented a 11-approximation algorithm for MinDisagree on complete bipartite graphs. The latter was subsequently improved by [1] to a factor of 4 and by [12] to a factor of 3.

In the context of Grothendieck's Inequality, numerous studies were conducted. For bounding K_G, Grothendieck [21] showed that $\frac{\pi}{2} \leq K_G \leq \sinh(\frac{\pi}{2})$. Later on, Reeds [29] presented a lower bound of 1.6769. The upper bound was also improved by Rietz [30] to 2.261 and by Krivine [26] to 1.78821 who conjectured that this is the correct number. Many years later, Braverman et al. [9] showed that Krivine's bound is not the correct answer for K_G, by slightly improving the upper bound. These days, the value of K_G is still unknown. Many extensions of Grothendieck's Inequality were presented and studied, e.g., [3, 25]. In addition, Nesterov [34] presented a $(2/\pi)$-approximation for Max Quad for the special case the matrix A is positive semi-definite.

2 Semi-Definite Relaxations for MaxCorr and Max-k-Corr

In this section we present natural semi-definite programming relaxations for both MaxCorr and Max-k-Corr. These relaxation apply to both general and bipartite graphs. Focusing first on MaxCorr, we consider the following natural semi-definite programming formulation, denoted by SDP, which assigns to each vertex u a unit vector y_u.

$$\begin{aligned} \max \quad & \sum_{(u,v)\in E^+} w_{u,v} \cdot (2y_u \cdot y_v - 1) + \sum_{(u,v)\in E^-} w_{u,v} \cdot (1 - 2y_u \cdot y_v) && (1)\\ \text{s.t.} \quad & y_v \cdot y_v = 1 && \forall v \in V \\ & y_u \cdot y_v \geq 0 && \forall u, v \in V \end{aligned}$$

In the following lemma, we prove that (1) is a relaxation[5] for the MaxCorr problem.

▶ **Lemma 3.** *Let* $\mathrm{OPT}_{\mathrm{SDP}}$ *be the optimum value of* (1)*, and* OPT *be the value of the optimal clustering. Then it holds that* $\mathrm{OPT}_{\mathrm{SDP}} \geq \mathrm{OPT}$.

[5] The integrality gap of this relaxation is discussed in Appendix C.

Proof. Given an optimal solution $\mathcal{C}_{\text{OPT}}$, we can assign each y_v to be a standard unit vector in $\mathbb{R}^n$, while a pair of vectors y_u, y_v will be the same unit vector if and only if u and v are in the same cluster in $\mathcal{C}_{\text{OPT}}$. This way, all the constrains are satisfied: for each $v \in V$ we have that $y_v \cdot y_v = 1$, and for each $u, v \in V$ it holds that $y_u \cdot y_v \in \{0, 1\}$. Moreover, the value of this solution for the SDP formulation is the same as $\text{Corr}(\mathcal{C}_{\text{OPT}})$: for a plus edge (u, v), if u and v are in the same cluster in $\mathcal{C}_{\text{OPT}}$, then $y_u \cdot y_v = 1$ and so it contributes $w_{u,v}$. If they are in different cluster, then $y_u \cdot y_v = 0$ and the edge contributes $-w_{u,v}$ to the objective function. Similar argument can be placed for the minus edges. Therefore, it holds that $\text{OPT}_{\text{SDP}} \geq \text{OPT}$. ◀

Recall that in Max-k-Corr the output can contain at most k clusters. Therefore, focusing on Max-k-Corr we consider the following natural semi-definite formulation, inspired by the work of Karger *et. al.* on 3-Coloring [23], which we denote by SDP_k:

$$\begin{aligned} \max \quad & \sum_{(u,v)\in E^+} w_{u,v}\left(\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k}\right) + \sum_{(u,v)\in E^-} w_{u,v}\left(\frac{k-2}{k} - \frac{2(k-1)}{k} y_u \cdot y_v\right) \\ \text{s.t.} \quad & y_v \cdot y_v = 1 && \forall v \in V \\ & y_u \cdot y_v \geq -1/(k-1) && \forall u, v \in V \end{aligned} \tag{2}$$

One may observe, that if we are limited to two clusters only, i.e., $k = 2$, then SDP_2 is exactly the standard semi-definite relaxation for both Max Quad and Max BiQuad (as in Grothendieck's inequality). Moreover, when $k \to \infty$, namely k is large enough such that Max-k-Corr becomes MaxCorr, SDP_∞ becomes the relaxation for MaxCorr. Similarly to MaxCorr, we prove the following lemma which shows that (2) is indeed a relaxation for the problem of Max-k-Corr.

▶ **Lemma 4.** *Let* $\text{OPT}_{\text{SDP}_k}$ *be the optimum value of* (2)*, and* OPT_k *be the value of the optimal clustering with at most* k *clusters. Then it holds that* $\text{OPT}_{\text{SDP}_k} \geq \text{OPT}_k$.

Proof. Given an optimal solution with at most k clusters, $\mathcal{C}_{\text{OPT}}$, we construct the following fractional solution. Let $v_1, \ldots, v_k$ be the vectors that form the $k-1$-regular simplex centered at the origin. We know that $v_1 + \cdots + v_k = 0$, and therefore

$$0 = \|v_1 + \cdots + v_k\|^2 = \sum_{i=1}^{k} v_i \cdot v_i + \sum_{i \neq j} v_i \cdot v_j = k + \sum_{i \neq j} v_i \cdot v_j.$$

Since the vertices of the regular simplex are in equal distance from each other, it must be the case that $v_i \cdot v_j = -1/(k-1)$ for all $i \neq j$. One can show that this is the most efficient solution, in terms of maximizing the minimum distance between k points on the Euclidean unit sphere.

Let us assign the vectors $\{y_v\}_{v\in V}$ to the vectors $v_1, \ldots, v_k$ where $y_u = y_v$ iff u and v are in the same cluster in $\mathcal{C}_{\text{OPT}}$. One can see that if $y_u \cdot y_v = 1$, it holds that

$$\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k} = \frac{2k-2-k+2}{k} = 1,$$

and if $y_u \cdot y_v = -1/(k-1)$, then it holds that

$$\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k} = \frac{-2-k+2}{k} = -1.$$

Hence, the value of the objective of (2) with this fractional solution equals the value of the integral solution. Thus, $\text{OPT}_{\text{SDP}_k} \geq \text{OPT}_k$. ◀

3 Approximation Algorithms for MaxCorr

In this section we present our approximation algorithms for MaxCorr on bipartite graphs, thus proving Theorem 1. First, for simplicity of presentation, we show that a straightforward adaptation of Krivine's method to SDP yields an approximation of 0.198. Second, we present an improved algorithm that achieves the guarantee of Theorem 1 by extending Krivine's method to more than two clusters.

3.1 Case Study: A Simple Two Clusters Algorithm

In this section we present a straightforward adaptation of Krivine's method to round SDP that produces at most two clusters. Denote by $V = V_1 \cup V_2$ the two sides of the graph and by n the total number of vertices in V. Let $A \in \mathbb{R}^{n \times n}$ be the matrix of the solution of (1). That is, $A_{i,j} = y_i \cdot y_j$ for all $i, j \in V$. Given a solution A, we can construct the corresponding vectors $\{y_v\}_{v \in V}$ in polynomial time. Additionally, we note that (1) can be solved, in polynomial time, and the value of the solution will be far from the optimum up to some additive error which is arbitrarily small.

Algorithm 1 Two Clusters Algorithm.

Input: $G = (V_1, V_2, E)$ and $w : E \to \mathbb{R}$
Output: clustering $\mathcal{C}$

1 Solve (1) for G and w to obtain a positive semi-definite matrix $A \in \mathbb{R}^{n \times n}$.
2 Define $\tilde{A} \in \mathbb{R}^{n \times n}$ as follows: $\tilde{A}_{i,j} \leftarrow \begin{cases} f(A_{i,j}) & \text{if } i \text{ and } j \text{ in different sides of } V \\ g(A_{i,j}) & \text{otherwise} \end{cases}$
3 Find vectors $\{\tilde{y}_i\}_{i \in V}$ s.t.: $\tilde{y}_i \cdot \tilde{y}_j = \tilde{A}_{i,j}$ $\forall i, j \in V$ (via Cholesky decomposition of $\tilde{A}$).
4 Choose z uniformly at random from S^{n-1}.
5 Set $C_1 \leftarrow \{i \in V : \tilde{y}_i \cdot z \geq 0\}$ and $C_2 \leftarrow \{i \in V : \tilde{y}_i \cdot z < 0\}$.
6 Return $\mathcal{C} = \{C_1, C_2\}$.

Next, we show how to choose the transformation functions f and g, and prove that the algorithm achieves the desired approximation factor. Specifically, we prove that it suffices to choose:

$$f(x) \triangleq \sin(c\pi x)\cos(\frac{c\pi}{2}) - \cos(c\pi x)\sin(\frac{c\pi}{2})$$
$$g(x) \triangleq \sinh(c\pi x)\cos(\frac{c\pi}{2}) + \cosh(c\pi x)\sin(\frac{c\pi}{2}),$$

where c is the solution to the equation $\sinh(c\pi)\cos(\frac{c\pi}{2}) + \cosh(c\pi)\sin(\frac{c\pi}{2}) = 1$. The reader is referred to Appendix A for the complete analysis.

3.2 Producing More Clusters – Better Approximation

Algorithm 1 always clusters the graph into at most two clusters while ignoring the fact that the optimal solution might contain a larger number of clusters. Building upon Algorithm 1, we present an algorithm that clusters the graph into at most three clusters. Though this seems like only a minor change, it enables us to improve the approximation factor to 0.254, well beyond the 0.2296 barrier of the approach of Charikar and Wirth [11].

We note that Algorithm 2 differs from Algorithm 1 since it uniformly at random picks one of the two initial clusters and splits it again using another independent random hyperplane. This subtle change incurs a significant change in the choice of f and g, as will be evident

Algorithm 2 Three Clusters Algorithm.

Input: $G = (V_1, V_2, E)$ and $w : E \to \mathbb{R}$
Output: clustering $\mathcal{C}$

Solve (1) for G and w to obtain a positive semi-definite matrix $A \in \mathbb{R}^{n \times n}$.
Define $\tilde{A} \in \mathbb{R}^{n \times n}$ as follows: $\tilde{A}_{i,j} \leftarrow \begin{cases} f(A_{i,j}) & \text{if } i \text{ and } j \text{ in different sides of } V \\ g(A_{i,j}) & \text{otherwise} \end{cases}$
Find vectors $\{\tilde{y}_i\}_{i \in V}$ s.t.: $\tilde{y}_i \cdot \tilde{y}_j = \tilde{A}_{i,j}\ \forall i, j \in V$ (via Cholesky decomposition of $\tilde{A}$).
Choose z and z' independently and uniformly from S^{n-1}.
Set $C' \leftarrow \{i \in V : \tilde{y}_i \cdot z \geq 0\}$ and $C'' \leftarrow \{i \in V : \tilde{y}_i \cdot z < 0\}$.
Let C_1 be a uniform random cluster from C' and C'', and C the remaining cluster.
Set $C_2 \leftarrow \{i \in C : \tilde{y}_i \cdot z' \geq 0\}$ and $C_3 \leftarrow \{i \in C : \tilde{y}_i \cdot z' < 0\}$.
Return $\mathcal{C} = \{C_1, C_2, C_3\}$.

from the analysis. Furthermore, this subtle change also makes the analysis more challenging, compared to Krivine's original method. The reason is that the separation probability of two vectors is quadractically dependent on the angle between them, as opposed to linear dependency in Krivine's original method. In order to analyze this algorithm, we first calculate the probability that two vertices will be in the same cluster.

▶ **Lemma 5.** *Let $u, v \in V$ whose corresponding vectors are $\tilde{y}_u, \tilde{y}_v$ respectively. Then,*

$$\Pr[u, v \text{ are in the same cluster}] = \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^2.$$

Proof. If u and v are in different clusters after splitting V into C' and C'', then u and v will remain apart. Otherwise, with probability $1/2$ we will choose to split their cluster, and then the chance of separating u and v is the same as before. That is,

$$\Pr[u, v \text{ are in the same cluster}] = \left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) \cdot \left(\frac{1}{2} + \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)\right).$$

This concludes the proof. ◀

Similarly to Lemma 11 we have the following lemma. In what follows, $X_{u,v}$ is the random variable denoting the contribution of the edge (u, v) to the value of the output of Algorithm 2.

▶ **Lemma 6.** *For every edge $(u, v) \in E$:*

$$\mathbb{E}[X_{u,v}] = \begin{cases} w_{u,v} \cdot \left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right) & (u, v) \in E^+ \\ w_{u,v} \cdot \left(3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2} - 1\right) & (u, v) \in E^-. \end{cases}$$

Proof. For every edge $(u, v) \in E^+$:

$$\begin{aligned} \mathbb{E}[X_{u,v}] &= w_{u,v} \cdot \Pr[u, v \text{ are in the same cluster}] - w_{u,v} \cdot (1 - \Pr[u, v \text{ are in the same cluster}]) \\ &= w_{u,v} \cdot \left(2\left(\frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^2\right) - 1\right) \\ &= w_{u,v} \cdot \left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right). \end{aligned}$$

For a minus edge, we can simply multiply the above by -1. ◀

The following lemma lies at the heart of the argument as it provides the existence of suitable transformations f and g.

▶ **Lemma 7.** *There exist functions $f, g : \mathbb{R} \to \mathbb{R}$ and a constant $c \in (0,1)$ such that the matrix $\tilde{A}$ obtained in Algorithm 2 can be decomposed into unit vectors $\{\tilde{y}_v\}_{v\in V}$ in $\mathbb{R}^n$, and these vectors satisfy that for all $(u,v) \in E$:*

$$\begin{cases} \left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right) = c \cdot (2y_u \cdot y_v - 1) & \text{if } (u,v) \in E^+ \\ \left(3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2} - 1\right) = c \cdot (1 - 2y_u \cdot y_v) & \text{if } (u,v) \in E^-. \end{cases}$$

Proof. We wish to find the appropriate functions f, g and the constant c, that will satisfy the above conditions. That is, we need to solve the equation:

$$\left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right) = c \cdot (2y_u \cdot y_v - 1)\,.$$

It has two solutions: $\arccos(\tilde{y}_u \cdot \tilde{y}_v) = \frac{1}{2}(3\pi \pm \pi\sqrt{5 - 4c + 8c(y_u \cdot y_v)})$. Therefore, we have two candidates for f: $\cos(\frac{1}{2}(3\pi - \pi\sqrt{5 - 4c + 8cx}))$ and $\cos(\frac{1}{2}(3\pi + \pi\sqrt{5 - 4c + 8cx}))$. We know that $\arccos(x)$ is bounded between 0 and π, and expect that f will satisfy $f(1) > f(0)$, so we dismiss the latter candidate solution and remain with:

$$f(x) = \cos(\frac{1}{2}(3\pi - \pi\sqrt{5 - 4c + 8cx})).$$

Moreover, since $y_u \cdot y_v \geq 0$ and $0 \leq c \leq 1$, it holds that $5 - 4c + 8c(y_u \cdot y_v) \geq 0$ for all u, v and so f is well defined for our case. In fact, we can see that f is well defined for all $x \geq {}^{(4c-5)}/_{(8c)}$. Hence, we consider the Taylor expansion of f at $x = 0$, defined by $p(x) = \sum_{k=0}^{\infty}(k!)^{-1} f^{(k)}(0)x^k$. Let f_k be the coefficient of x^k in $p(x)$. We note that the coefficients depend on the value of c. Thus, $p(x)$ has the following two properties, described in two technical lemmas that are given without a proof.

▶ **Lemma 8** (The convergence radius of p)**.** *The function $f(x)$ is equal to its Taylor series, $p(x)$, for all $x \in (\frac{4c-5}{8c}, -\frac{4c-5}{8c})$, and in particular, $p(x) = f(x)$ for all $x \in [-1,1]$ if $c < 5/12$.*

▶ **Lemma 9** (The signs of the coefficients of p)**.** *For all $c \in (0.25, 0.4)$, it holds that $f_0 < 0$, $f_1 > 0$, $f_2 < 0$ and for all $k \geq 2$, $f_{2k} > 0$ and $f_{2k-1} < 0$.*

Thus, from now on we assume that c is in the range $(0.25, 0.4)$, and can consider the function $f(-x)$. It holds that for all $x \in [-1,1]$,

$$f(-x) = p(-x) = \sum_{k=0}^{\infty} f_{2k}x^{2k} - \sum_{k=0}^{\infty} f_{2k+1}x^{2k+1} = f_0 - f_1 x + f_2 x^2 + \sum_{k=3}^{\infty} |f_k| x^k.$$

Therefore, we conclude that $\sum_{k=0}^{\infty} |f_k| x^k = f(-x) - 2f_0 + 2f_1 x - 2f_2 x^2$. Hence, let us define $g(x) = f(-x) - 2f_0 + 2f_1 x - 2f_2 x^2$. We can prove that $\tilde{A}$ is positive semi-definite, and can be decomposed to the above vectors $\{\tilde{y}_v\}_{v\in V}$ in a similar way to Lemma 13. Next, we wish to find the value of c for which the new vectors $\{\tilde{y}_v\}_{v\in V}$ will be unit vectors. Plugging $x = 1$ in the equation $g(x) = 1$ and then solving for c in the required range, will yield the solution $c \approx 0.254013$. This completes the proof of Lemma 7. ◀

Proof of Theorem 1. According to Lemma 6 and Lemma 7, it follows that Algorithm 2 achieves an approximation of 0.254 for the problem of MaxCorr on bipartite graphs. ◀

As a result from Theorem 1, we obtain an upper bound on α_∞, proving the second part of Theorem 2. Additionally, we note that Algorithm 2 can be slightly improved. If we consider the probability that two vertices will be in the same cluster, we can notice that it is the same as if we split the hyper-sphere with one hyper-plane with probability 1/2, or with two hyper-planes with probability 1/2. Hence, we can find the value of p for which the algorithm that splits the hyper-sphere with one hyper-plane with probability p or with two hyper-planes with probability $1-p$, achieves the best approximation factor (with the above analysis). It turns out that the best p is very close to 1/2, i.e., if we set $p = 0.49$ the approximation factor will be $c \approx 0.2551$. As previously mentioned, allowing the algorithm to produce more than four clusters is not beneficial (refer to Appendix B for additional details).

4 Restricting the Number of Clusters – Approximating Max-k-Corr

In this section we focus on Max-k-Corr, where the solution is constrained to contain at most k clusters. We build upon Algorithms 1 and 2 to obtain our results.

4.1 Two Clusters – Adapting Algorithm 1

We start with adapting Algorithm 1 to Max-k-Corr. It is obvious that this algorithm produces a feasible solution, no matter which transformations f and g we employ, since $k \geq 2$ and it produces at most two clusters. However, since SDP_k changes with k, the original transformations f and g, that were chosen when considering MaxCorr and SDP (equivalently SDP_∞), do not yield a meaningful approximation anymore. Thus, all that remains is to choose these transformations, given k. This is done by requiring that the transformed vectors satisfy the following for every edge $(u,v) \in E$:

$$1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} = c\left(\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k}\right).$$

To satisfy the above we choose f as follows:

$$\begin{aligned} f(x) &= \cos\left(\frac{\pi}{2k}(k + c(k-2) - 2cx(k-1))\right) \\ &= \sin\left(c\pi x\frac{k-1}{k}\right)\cos\left(\frac{c\pi}{2}\frac{k-2}{k}\right) - \cos\left(c\pi x\frac{k-1}{k}\right)\sin\left(\frac{c\pi}{2}\frac{k-2}{k}\right). \end{aligned}$$

Hence, the function g will be:

$$g(x) = \sinh\left(c\pi x\frac{k-1}{k}\right)\cos\left(\frac{c\pi}{2}\frac{k-2}{k}\right) + \cosh\left(c\pi x\frac{k-1}{k}\right)\sin\left(\frac{c\pi}{2}\frac{k-2}{k}\right).$$

The analysis continues the same as in Algorithm 1, and we require for the transformed vectors to be unit vectors, i.e., that $g(1) = 1$. Solving this equation for each value of k, yields the desired approximation factor c. These appear in Table 2 for some values of k.

Table 2 Approximation factors obtained by adapting Algorithm 1 for Max-k-Corr.

k	2	3	4	5	10	50	1000
c	0.5611	0.3441	0.2901	0.2654	0.2269	0.2034	0.1985

We can see that when $k = 2$, we have exactly the constant obtained by Krivine. As k increases the approximation factor gracefully degrades, and when $k \to \infty$ it approaches 0.1985 (the approximation we obtained for MaxCorr with Algorithm 1). Additionally, we note that this result proves the first part of Theorem 2 as it provides upper bounds on α_k for $k \geq 2$.

4.2 More Than Two Clusters – Adapting Algorithm 2

The success of Algorithm 2 implies that allowing more than two clusters in the output might be beneficial. However, adapting this approach, and specifically Algorithm 2, to Max-k-Corr requires much care as the value of k will have a considerable influence on the resulting algorithm and its transformations.

Specifically, given k, we need to choose from the following clustering techniques: (1) two clusters produced by a random hyperplane (similarly to Algorithm 1); (2) three clusters produced by a random hyperplane and then further partioning of one of the clusters by an additional random hyperplane (similarly to Algorithm 2); and (3) four clusters produced by choosing two random hyperplanes. The exact convex combination of these techniques will uniquely dictate the transformations f and g that will ensure the success of the algorithm. We refer the reader to Appendix B for a more detailed discussion of the above, the approximation guarantees obtained for specific values of k, and additional details, e.g., why not use a larger number of hyperplanes to construct a clustering.

References

1 Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. Improved approximation algorithms for bipartite correlation clustering. *SIAM Journal on Computing*, 41(5):1110–1121, 2012.

2 Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)*, 55(5):23, 2008.

3 Noga Alon, Konstantin Makarychev, Yury Makarychev, and Assaf Naor. Quadratic forms on graphs. *Inventiones mathematicae*, 163(3):499–522, 2006.

4 Noga Alon and Assaf Naor. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 72–80, 2004.

5 Noga Amit. *The bicluster graph editing problem.* PhD thesis, Tel Aviv University, 2004.

6 Megasthenis Asteris, Anastasios Kyrillidis, Dimitris Papailiopoulos, and Alexandros Dimakis. Bipartite correlation clustering: Maximizing agreements. In *Artificial Intelligence and Statistics*, pages 121–129, 2016.

7 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

8 Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of computational biology*, 6(3-4):281–297, 1999.

9 Mark Braverman, Konstantin Makarychev, Yury Makarychev, and Assaf Naor. The grothendieck constant is strictly smaller than krivine's bound. ieee 52nd annual symposium on foundations of computer science focs 2011, 453–462. *IEEE Computer Soc., Los Alamitos, CA*, 2011.

10 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

11 Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60. IEEE, 2004.

12 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal lp rounding algorithm for correlationclustering on complete and complete k-partite graphs. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 219–228, 2015.

13 Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

14 William Cohen and Jacob Richman. Learning to match and cluster entity names. In *In ACM SIGIR-2001 Workshop on Mathematical/Formal Methods in Information Retrieval*, 2001.

15 William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 475–480, New York, NY, USA, 2002. Association for Computing Machinery.

16 Tom Coleman, James Saunderson, and Anthony Wirth. A local-search 2-approximation for 2-correlation-clustering. In *European Symposium on Algorithms*, pages 308–319. Springer, 2008.

17 Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

18 Vladimir Filkov and Steven Skiena. Integrating microarray data by consensus clustering. *International Journal on Artificial Intelligence Tools*, 13(04):863–880, 2004.

19 Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-cut and MAX BISECTION. *Algorithmica*, 18(1):67–81, 1997. doi:10.1007/BF02523688.

20 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(13):249–266, 2006. doi:10.4086/toc.2006.v002a013.

21 Alexandre Grothendieck. *Résumé de la théorie métrique des produits tensoriels topologiques*. Soc. de Matemática de São Paulo, 1956.

22 Venkatesan Guruswami and Prasad Raghavendra. Constraint satisfaction over a non-boolean domain: Approximation algorithms and unique-games hardness. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 77–90. Springer, 2008.

23 David R. Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998. doi:10.1145/274787.274791.

24 Marek Karpinski and Warren Schudy. Linear time approximation schemes for the gale-berlekamp game and related minimization problems. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 313–322, 2009.

25 Subhash Khot and Assaf Naor. Grothendieck-type inequalities in combinatorial optimization. *arXiv preprint*, 2011. arXiv:1108.2464.

26 Jean-Louis Krivine. Constantes de grothendieck et fonctions de type positif sur les spheres. *Séminaire Analyse fonctionnelle (dit" Maurey-Schwartz")*, pages 1–17, 1978.

27 Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45, 2004.

28 Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, August 2003.

29 JA Reeds. A new lower bound on the real grothendieck constant. *Manuscript (http://www. dtc. umn. edu/ reedsj/bound2. dvi)*, 88(96):107, 1991.

30 Ronald E Rietz. A proof of the grothendieck inequality. *Israel journal of mathematics*, 19(3):271–276, 1974.

31 Cornelis Roos, Tamás Terlaky, Arkadi Nemirovski, and Kees Roos. On maximization of quadratic form over intersection of ellipsoids with common center. *Mathematical Programming*, 86, September 1999.

32 Chaitanya Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 526–527. Society for Industrial and Applied Mathematics, 2004.

33 Jurgen Van Gael and Xiaojin Zhu. Correlation clustering for crosslingual link detection. In *IJCAI*, pages 1744–1749, 2007.

34 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: `http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE`.

35 Anthony Wirth. Correlation clustering. *C. Sammut and G. Webb, editors, Encyclopedia of Machine Learning*, pages 227—-231, 2010.

36 Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32, 2001.

A Analysis of Algorithm 1

Lemma 10 is widely known and is given without proof (see, e.g., [3]).

▶ **Lemma 10.** *Let y_u, y_v be two unit vectors in $\mathbb{R}^n$ and z be a random unit vector chosen uniformly from S^{n-1}. Then,*

$$\Pr[\text{sign}(y_u \cdot z) = \text{sign}(y_v \cdot z)] = 1 - \frac{\arccos(y_u \cdot y_v)}{\pi}.$$

▶ **Lemma 11.** *Let $\tilde{y}_u, \tilde{y}_v$ be the vectors representing u and v respectively. Additionally, denote by $X_{u,v} = w_{u,v} \cdot \text{sign}(\tilde{y}_u \cdot z)\text{sign}(\tilde{y}_v \cdot z)$ the random variable that represents the contribution of the edge (u, v) to the value of the solution. Therefore,*

$$\mathbb{E}[X_{u,v}] = \begin{cases} w_{u,v}\left(1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) & (u,v) \in E^+ \\ w_{u,v}\left(2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1\right) & (u,v) \in E^-. \end{cases}$$

Proof. From Algorithm 1, we can see that that two vertices u, v will be in the same cluster if and only if $\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)$. Therefore, for every plus edge $(u, v) \in E^+$,

$$\begin{aligned} \mathbb{E}[X_{u,v}] &= w_{u,v} \Pr[\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)] + (-1)w_{u,v}(1 - \Pr[\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)]) \\ &= w_{u,v}(2 \Pr[\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)] - 1). \end{aligned}$$

Additionally, for every minus edge $(u, v) \in E^-$,

$$\begin{aligned} \mathbb{E}[X_{u,v}] &= w_{u,v}(1 - \Pr[\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)]) + (-1)w_{u,v} \Pr[\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)] \\ &= w_{u,v}(1 - 2 \Pr[\text{sign}(\tilde{y}_u \cdot z) = \text{sign}(\tilde{y}_v \cdot z)]) \end{aligned}$$

Consequently, we can use Lemma 10 to complete this proof. ◀

The following corollary is immediate from linearity of expectation.

▶ **Corollary 12.** *Let X_{ALG} be the value of the output of Algorithm 1 . Then*

$$\mathbb{E}[X_{ALG}] = \sum_{(u,v)\in E^+} w_{u,v}\left(1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \sum_{(u,v)\in E^-} w_{u,v}\left(2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1\right).$$

Proof. Follows directly from Lemma 11 and the linearity of expectation. ◀

The following lemma lies at the heart of the argument as it proves the existence of suitable transformations f and g.

▶ **Lemma 13.** *There exist functions $f, g : \mathbb{R} \to \mathbb{R}$ and a constant $c \in (0,1)$ such that the matrix $\tilde{A}$ obtained in Algorithm 1 can be decomposed into unit vectors $\{\tilde{y}_v\}_{v \in V}$ in $\mathbb{R}^n$, and these vectors satisfy that for all $(u,v) \in E^+$,*

$$1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} = c(2y_u \cdot y_v - 1)$$

and for all $(u,v) \in E^-$,

$$2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1 = c(1 - 2y_u \cdot y_v).$$

Proof. First, we want to choose a function f such that

$$1 - 2\frac{\arccos(f(y_u \cdot y_v))}{\pi} = c(2y_u \cdot y_v - 1)$$

for all $(u,v) \in E$. This will satisfy the condition on plus edges and minus edges as well. The constant c will be determined later. We choose the function f in the following way:

$$f(x) \triangleq \cos\left(\frac{\pi}{2}(1 - c(2x-1))\right) = \sin\left(c\pi x - \frac{c\pi}{2}\right) = \sin(c\pi x)\cos(\frac{c\pi}{2}) - \cos(c\pi x)\sin(\frac{c\pi}{2}),$$

where the last equality follows from the identity: $\sin(a-b) = \sin(a)\cos(b) - \cos(a)\sin(b)$ for all $a, b \in \mathbb{R}$. In addition, using the Taylor series expansion of $\sin(x)$ and $\cos(x)$, f can be written as follows:

$$f(x) = \cos(\frac{c\pi}{2}) \sum_{k=0}^{\infty} \frac{(-1)^k (c\pi x)^{2k+1}}{(2k+1)!} - \sin(\frac{c\pi}{2}) \sum_{k=0}^{\infty} \frac{(-1)^k (c\pi x)^{2k}}{(2k)!}.$$

Let f_k be the coefficient of x^k in the Taylor expansion of f. We define the following functions:

$$g(x) = \sum_{k=0}^{\infty} |f_k| x^k, \qquad a(x) = \sum_{k=0}^{\infty} \sqrt{|f_k|} x^k, \qquad b(x) = \sum_{k=0}^{\infty} \text{sign}(f_k)\sqrt{|f_k|} x^k.$$

Let a_k and b_k be the coefficients of x^k in the above series expansions of a and b. Recall that our goal is to show that there exist vectors $\{\tilde{y}_v\}_{v \in V}$ such that $\tilde{y}_u \cdot \tilde{y}_v = \tilde{A}_{u,v}$ for all $u, v \in V$. Given the set of vectors $\{y_v\}_{v \in V}$, we define the following vectors: if $u \in V_1$ then

$$y'_u = (a_0, a_1 y_u, a_2(y_u \otimes y_u), a_3(y_u \otimes y_u \otimes y_u), \dots)$$

and if $u \in V_2$ then

$$y'_u = (b_0, b_1 y_u, b_2(y_u \otimes y_u), b_3(y_u \otimes y_u \otimes y_u), \dots).$$

We note that the k-times tensor product $y_u \otimes y_u \otimes \cdots \otimes y_u$ are in fact n^k coordinates in the vector y'_u, as n is the dimension of the vector y_u. For every vector v and integer k, we denote by $v^{\otimes k}$ the k-times tensor product of v with itself. It is a known fact, that for every two vectors u, v and integer k, it holds that $u^{\otimes k} \cdot v^{\otimes k} = (u \cdot v)^k$.

Thus, we can see that if u, v are not both in V_1 or V_2, that is, not in the same side of the bipartite graph, then

$$y'_u \cdot y'_v = \sum_{k=0}^{\infty} a_k b_k (y_u^{\otimes k} \cdot y_v^{\otimes k}) = \sum_{k=0}^{\infty} a_k b_k (y_u \cdot y_v)^k = \sum_{k=0}^{\infty} f_k (y_u \cdot y_v)^k = f(y_u \cdot y_v).$$

Otherwise, if u and v are in the same side of the graph, without loss of generality, let it be V_1, then:

$$y'_u \cdot y'_v = \sum_{k=0}^{\infty} a_k^2 (y_u^{\otimes k} \cdot y_v^{\otimes k}) = \sum_{k=0}^{\infty} |f_k| (y_u \cdot y_v)^k = g(y_u \cdot y_v).$$

That is, it holds that $\tilde{A}_{i,j} = y'_i \cdot y'_j$ for all $i, j \in V$. Consequently, the matrix $\tilde{A}_{i,j} \in \mathbb{R}^{n \times n}$ is the symmetric Gramian matrix of the vectors $\{y'_v\}_{v \in V}$, and can be decomposed in polynomial time with Cholesky decomposition into $\tilde{A} = \tilde{Y}\tilde{Y}^T$. The rows of $\tilde{Y}$ will be the desired vectors $\{\tilde{y}_v\}_{v \in V}$.

To complete the proof we need to show that these vectors are unit vectors. That is, we want to show that $y'_u \cdot y'_u = 1$ for all $u \in V$. Since y_u is a unit vector, we have that

$$y'_u \cdot y'_u = \sum_{k=0}^{\infty} |f_k| (y_u \cdot y_v)^k = g(1).$$

Since $c \in (0, 1)$, we have that $\cos(\frac{c\pi}{2}) \geq 0$ and $\sin(\frac{c\pi}{2}) \geq 0$. Therefore, we can write g in the form:

$$g(x) = \sum_{k=0}^{\infty} |a_k| x^k = \sinh(c\pi x) \cos(\frac{c\pi}{2}) + \cosh(c\pi x) \sin(\frac{c\pi}{2}).$$

Hence, we can choose c to be the solution of the equation

$$\sum_{k=0}^{\infty} |f_k| = \sinh(c\pi) \cos(\frac{c\pi}{2}) + \cosh(c\pi) \sin(\frac{c\pi}{2}) = 1$$

in the range $(0, 1)$, and the vectors $\{\tilde{y}_v\}_{v \in V}$ will be unit vectors. The constant c turns out to be at least 0.19829. ◀

Finally, we are ready to complete the analysis of Algorithm 1, which is given in the following theorem.

▶ **Theorem 14.** *There exists a polynomial-time* 0.198*-approximation algorithm for the problem of* MaxCorr *on bipartite graphs that clusters the graph to at most 2 clusters.*

Proof of Theorem 14. From Corollary 12, Lemma 13 and Lemma 3, it follows that:

$$\begin{aligned}
\mathbb{E}[X_{ALG}] &= \sum_{(u,v) \in E^+} w_{u,v} \left(1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \sum_{(u,v) \in E^-} w_{u,v} \left(2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1\right) \\
&= \sum_{(u,v) \in E^+} w_{u,v} c(2y_u \cdot y_v - 1) + \sum_{(u,v) \in E^-} w_{u,v} c(1 - 2y_u \cdot y_v) \\
&= c \cdot \mathrm{OPT}_{\mathrm{SDP}} \geq c \cdot \mathrm{OPT},
\end{aligned}$$

where c is at least 0.19828. ◀

B Max-k-Corr: Adapting Algorithm 2

As previously mentioned, given k, we need to present the combination of clustering techniques that will be used to approximate Max-k-Corr. Let us start by doing exactly this. In what follows, it is important to note that given k: (1) the mixing probability p appearing below depends on k; and (2) the transformations f and g also depend on k.

First, when $k = 3$, with a probability of p we produce two clusters by a single random hyperplane and with the remaining probability of $1-p$ we produce three clusters as described in Algorithm 2 (use one random hyperplane to produce two clusters, choose uniformly at random one of the two clusters, and further partition it using the second random hyperplane). Second, when $k \geq 4$, with a probability of p we produce two clusters by a single random hyperplane and with the remaining probability of $1-p$ we produce four clusters by using two random hyperplanes.

Let focus now on the analysis, specifically, how the transformations f and g are chosen. First, when $k = 3$, the probability that two vertices u and v are in the same cluster equals:

$$p\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+(1-p)\left(\frac{1}{2}\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+\frac{1}{2}\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)^2\right)$$
$$=\frac{1}{2}(1+p)\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+\frac{1}{2}(1-p)\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)^2.$$

Thus, we wish to find the optimal parameters p and c that will satisfy that:

$$2\Pr[u,v \text{ are in the same cluster}]-1=c\left(\frac{4}{3}y_u\cdot y_v-\frac{1}{3}\right)$$

and the transformed vectors remain unit vectors. Hence, the function f is chosen to be:

$$f(x)=\cos\left(\frac{\pi\left(\sqrt{9\left(p^2-2p+5\right)-12c(p-1)(4x-1)}+3p-9\right)}{6(p-1)}\right),$$

and g is chosen accordingly. Performing the computation of the parameters, similarly to the previous sections of the paper, we obtain that $p\approx 0.47$ and $c\approx 0.397$.

Second, when $k \geq 4$, the probability that two vertices u and v are in the same cluster equals:

$$p\cdot\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+(1-p)\cdot\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)^2.$$

Thus, following the same footsteps as before we can conclude that f is chosen to be:

$$f(x)=\cos\left(\frac{\pi\left(\sqrt{-\frac{2c(p-1)(2kx-k-2x+2)}{k}+p^2-2p+2}+p-2\right)}{2(p-1)}\right),$$

and g is chosen accordingly. For every $k \geq 4$ we can repeat the process and find the best parameters p and c. Summarizing, we present these parameters (optimized numerically) for several values of k (refer to Table 3).

Table 3 Approximation factors obtained by adapting Algorithm 1 and Algorithm 2 for Max-k-Corr.

k	3	4	5	6	10
p	0.47	0.655	0.648	0.595	0.541
approximation	0.397	0.348	0.32	0.309	0.285

We note that when k is large it is not clear why one should not use more than two hyperplanes and obtain a clustering that might contain a large number of clusters. An interesting phenomenon arises when we want to split the hypersphere to m slices. If $m \in (2^t, 2^{t+1})$ for some integer t, then we can sample t random hyperplanes and split the hypersphere into 2^t slices. Then, we randomly choose $m - 2^t$ of the slices, and split each one of them with a random hyperplane. The result is m slices, and the probability that two vertices will be in the same cluster equals:

$$\frac{2^{t+1} - m}{2^t} \cdot \left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^t + \frac{m - 2^t}{2^t} \cdot \left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^{t+1}.$$

That is, this probability is a convex combination of the probabilities that two vertices will be in the same cluster when we randomly chose t or $t+1$ hyperplanes. Since the analysis of the algorithm is edge-wise, we can simply consider a distribution $\{p_t\}_{t>0}, \sum_{t=1}^{\infty} p_t = 1$, where p_t is the probability that we split the hypersphere into 2^t slices using t random independent hyperplanes. However, numeric calculations of splitting into more than four clusters did not yield better approximation factors. Thus, for a fixed k, the behavior of the value of c is unimodal, and reaches a peak when $p_1 + p_2 = 1$. Therefore, we focus on on algorithms that pick one hyperplane with a probability of p and two hyperplanes with the remaining probability of $1 - p$.

C The Integrality Gap of (1)

In this section, we wish to discuss the integrality gap of the relaxation presented in (1). We consider the following simple example: a 4-cycle graph that has 3 plus edges and 1 minus edge. We denote it by $V = \{1, 2, 3, 4\}$ and $E^+ = \{(1, 2), (2, 3), (3, 4)\}, E^- = \{(1, 4)\}$. Clearly, this graph is bipartite and the value of the optimal solution is 2. However, the optimum of (1) is obtained when the vectors y_1, y_4 are orthogonal, and $y_1 \cdot y_2 = y_2 \cdot y_3 = y_3 \cdot y_4 = \sqrt{3}/2$. One possible solution will be $y_1 = (1, 0), y_2 = (\sqrt{3}/2, 1/2), y_3 = (1/2, \sqrt{3}/2), y_4 = (0, 1)$, and the value of this solution is $3\sqrt{3} - 2$. Therefore, (1) admits an integrality gap of at most $\frac{2}{3\sqrt{3}-2} \approx 0.62575$.

In addition, we can give better bound on the integrality gap for general graphs. We consider the simple 3-cycle graph, that has two plus edges, $(1, 2), (2, 3)$, and one minus edge, $(1, 3)$. One can see that the value of the optimal integral solution is 1, whereas the fractional optimum is obtained when $y_1 \cdot y_3 = 0$ and $y_1 \cdot y_2 = y_2 \cdot y_3 = \sqrt{2}/2$. Therefore, the integrality gap on general graphs is at most $\frac{1}{2\sqrt{2}-1} \approx 0.54691$.

Streaming Complexity of SVMs

Alexandr Andoni
Columbia University, New York, NY, USA
andoni@cs.columbia.edu

Collin Burns
Columbia University, New York, NY, USA
collin.burns@columbia.edu

Yi Li
Nanyang Technological University, Singapore, Singapore
yili@ntu.edu.sg

Sepideh Mahabadi
Toyota Technological Institute at Chicago, IL, USA
mahabadi@ttic.edu

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@cs.cmu.edu

Abstract

We study the space complexity of solving the bias-regularized SVM problem in the streaming model. In particular, given a data set $(x_i, y_i) \in \mathbb{R}^d \times \{-1, +1\}$, the objective function is $F_\lambda(\theta, b) = \frac{\lambda}{2}\|(\theta, b)\|_2^2 + \frac{1}{n}\sum_{i=1}^n \max\{0, 1 - y_i(\theta^T x_i + b)\}$ and the goal is to find the parameters that (approximately) minimize this objective. This is a classic supervised learning problem that has drawn lots of attention, including for developing fast algorithms for solving the problem approximately: i.e., for finding (θ, b) such that $F_\lambda(\theta, b) \leq \min_{(\theta', b')} F_\lambda(\theta', b') + \varepsilon$.

One of the most widely used algorithms for approximately optimizing the SVM objective is Stochastic Gradient Descent (SGD), which requires only $O(\frac{1}{\lambda\varepsilon})$ random samples, and which immediately yields a streaming algorithm that uses $O(\frac{d}{\lambda\varepsilon})$ space. For related problems, better streaming algorithms are only known for *smooth* functions, unlike the SVM objective that we focus on in this work.

We initiate an investigation of the space complexity for both finding an approximate optimum of this objective, and for the related "point estimation" problem of sketching the data set to evaluate the function value F_λ on any query (θ, b). We show that, for both problems, for dimensions $d = 1, 2$, one can obtain streaming algorithms with space polynomially smaller than $\frac{1}{\lambda\varepsilon}$, which is the complexity of SGD for strongly convex functions like the bias-regularized SVM [12], and which is known to be tight in general, even for $d = 1$ [1]. We also prove polynomial lower bounds for both point estimation and optimization. In particular, for point estimation we obtain a tight bound of $\Theta(1/\sqrt{\varepsilon})$ for $d = 1$ and a nearly tight lower bound of $\widetilde{\Omega}(d/\varepsilon^2)$ for $d = \Omega(\log(1/\epsilon))$. Finally, for optimization, we prove a $\Omega(1/\sqrt{\epsilon})$ lower bound for $d = \Omega(\log(1/\epsilon))$, and show similar bounds when d is constant.

2012 ACM Subject Classification Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Machine learning theory; Theory of computation → Lower bounds and information complexity

Keywords and phrases support vector machine, streaming algorithm, space lower bound, sketching algorithm, point estimation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.50

Category APPROX

Related Version https://arxiv.org/abs/2007.03633



© Alexandr Andoni, Collin Burns, Yi Li, Sepideh Mahabadi, and David P. Woodruff;
licensed under Creative Commons License CC-BY

Funding *Alexandr Andoni*: Supported in part by Simons Foundation (#491119) and NSF (CCF-1617955, CCF- 1740833).
Yi Li: Supported in part by Singapore Ministry of Education (AcRF) Tier 2 grant MOE2018-T2-1-013.
David P. Woodruff: Supported by the National Science Foundation under Grant No. CCF-1815840.

1 Introduction

The Support Vector Machine (SVM) optimization problem is a classic supervised learning problem with a rich and extensive literature. In this work, we consider the SVM problem in the *space-constrained* setting. Specifically, we focus on the bias-regularized SVM[1]. For n labelled data points $(x_i, y_i) \in \mathbb{R}^d \times \{-1, +1\}$, with $\|x_i\| \leq 1$, and $(\theta, b) \in \mathbb{R}^d \times \mathbb{R}$ the unknown model parameters, the SVM objective function is defined as:

$$F_\lambda(\theta, b) := \frac{\lambda}{2}\|(\theta, b)\|_2^2 + \frac{1}{n}\sum_{i=1}^{n} \max\{0, 1 - y_i(\theta^T x_i + b)\}, \tag{1}$$

where λ is the regularization parameter. The SVM optimization problem is then to minimize the objective:

$$(\theta^*, b^*) := \arg\min_{\theta, b} F_\lambda(\theta, b). \tag{2}$$

This problem is of both theoretical and practical interest, and has received lots of attention in the machine learning community. One of the main lines of work on SVMs has focused on trying to find approximately optimal solutions quickly (see, e.g., [12], [3], [11], and the references therein). Most notably, using a variant of stochastic gradient descent (SGD), one can compute a solution $(\hat{\theta}, \hat{b})$ which is at most ε away from the optimal $F_\lambda(\theta^*, b^*)$ in $O(\frac{1}{\lambda\varepsilon})$ SGD steps, each using a single randomly sampled data point (x_i, y_i) [12].

However, in many applications of SVMs, the number of data points is sufficiently large that even storing all of the data may be prohibitively expensive. In this case, it may be desirable to store a smaller summary of the data that is alone sufficient for optimizing the SVM objective within a desired error tolerance.

This goal has been studied for related smooth objective functions, but not for non-smooth objectives like SVMs. For example, [5] focuses on this problem for the more general setting of Generalized Linear Models (GLMs), where F is defined as $F = \frac{1}{n}\sum_{i=1}^{n} \phi(y_i, \theta^T x_i)$ for an arbitrary function ϕ. This includes the SVM objective. The authors of [5] show that if ϕ is well-approximable by a low-degree polynomial, then one can stream through the data points while keeping a small sketch of the data that is sufficient for minimizing the objective. However, the space complexity depends exponentially on the degree of the approximating polynomial, so it is only feasible for relatively smooth functions ϕ.

In this work we study SVMs, the most common non-smooth GLM, and focus on space complexity in the streaming setting. In addition to optimization, we also focus on the problem of *point estimation*: sketching the data points so that, given (θ, b), we can output a value within ε of $F_\lambda(\theta, b)$. While we use this as an intermediate step for achieving improved optimization upper bounds in the low-dimensional setting, this problem is also of independent interest. It occurs, for example, in estimating the GLM posterior distribution: the distribution of the parameters (θ, b) given the observed data and some prior distribution over (θ, b) [5].

[1] In the standard SVM formulation, the bias is not regularized, but the bias-regularized version is common both in theoretical work and in practice. See, for example, [12].

1.1 Our Results

Our results are for the two considered problems, and include both upper and lower bounds. We present the results for the point estimation problem independently. Then we present our results for optimization, the upper bounds for which rely on our point estimation results.

Point estimation. First, we show that one can obtain a *multiplicative* $(1+\varepsilon)$-factor approximation for $d = 1$ that uses $\tilde{O}(1/\varepsilon^2)$ space. However, we then show that it is not possible to get a multiplicative approximation algorithm for dimension $d > 1$ that uses space sub-linear in n. Given this, we otherwise focus on the space complexity of *additive* $\pm\varepsilon$ approximation streaming algorithms. For $d = 1$, we obtain space $O(1/\sqrt{\varepsilon})$, and for $d = 2$, we obtain space $O(\varepsilon^{-4/5})$. We complement our algorithms with a lower bound of $\Omega(\varepsilon^{-(d+1)/(d+3)})$ for dimension d, for any sketching algorithm, which goes up to ε^{-1} as we increase d. Note that, for $d = 1$, our bounds are tight. For $d = 2$, our lower bound translates to $\Omega(\varepsilon^{-3/5})$.

We also prove a lower bound of $\Omega(d/(\varepsilon^2 \operatorname{polylog}(1/\varepsilon)))$ for $d = \Omega(\log(1/\varepsilon))$, which is tight up to $\operatorname{polylog}(1/\varepsilon)$ factors. Together with the $O(1/\lambda\varepsilon)$ upper bound achieved from SGD, this shows that there is a *strict gap* between point estimation and optimization. This is also the case for linear regression in the streaming model: while getting a multiplicative $1 \pm \varepsilon$ point estimation approximation for linear regression requires space $\tilde{\Theta}(\varepsilon^{-2})$ [6], the (multiplicative) optimization problem requires only $\tilde{\Theta}(\varepsilon^{-1})$ space [4].

Optimization. First, using standard net arguments, we show how to use our point estimation results to approximately minimize the SVM objective.

▶ **Theorem 1.** *Suppose there is a streaming algorithm that, after seeing data* $\{(x_i, y_i)\}_{i=1}^n$, *where* $\|x_i\| \le 1$, *can produce a sketch of size* s *that, given any* (θ, b) *such that* $\|(\theta, b)\| \le \sqrt{2/\lambda}$, *is able to output* $\hat{F}(\theta, b)$ *such that* $|\hat{F}(\theta, b) - F(\theta, b)| \le \varepsilon$ *with probability at least* 0.9. *Then there is also a streaming algorithm that, under the same input, will output* $(\hat{\theta}, \hat{b})$ *with* $|F_\lambda(\hat{\theta}, \hat{b}) - F_\lambda(\theta^*_\lambda, b^*_\lambda)| \le \varepsilon$ *with probability at least* 0.9, *while using space* $O(s \cdot d \log d/(\lambda\varepsilon))$.

Together with our point estimation results, we immediately obtain streaming algorithms that, for dimensions one and two, obtain space polynomially smaller than $\frac{1}{\lambda\varepsilon}$, which is the complexity of SGD for strongly convex functions like the bias-regularized SVM [12], and which is tight in general [1].

We also prove space lower bounds for optimization. First, we consider the high-dimensional case when $\lambda = 10^{-4}$ is a constant:

▶ **Theorem 2.** *Let* $\lambda = 10^{-4}$ *and* $d = O(\log n)$ *for* $n = \Theta(\varepsilon^{-1/2})$. *Suppose there exists a sketch such that, given a stream of inputs* $\{(x_i, y_i)\}_{i=1}^n$, *outputs some* $(\hat{\theta}, \hat{b})$ *with probability at least* 0.9 *such that* $F_\lambda(\hat{\theta}, \hat{b}) \le F_\lambda(\theta^*, b^*) + \varepsilon$. *Then such a sketch requires* $\Omega(\varepsilon^{-1/2})$ *space.*

We can adapt the lower bound to the low dimensional setting if we let $\lambda = \operatorname{poly}(1/n)$. Specifically, we show a somewhat weaker lower bound of $\Omega(\varepsilon^{-1/4})$ for $d = 2$ as long as $\lambda = \Theta(1/n^2)$. Moreover, we are able to show the same $\Omega(\varepsilon^{-1/2})$ bound for any $d \ge 3$ as long as $\lambda = \Theta(1/n)$. Note that $\lambda = 1/n$ is a reasonable setting that is often used in practice.

1.2 Related Work

Stochastic optimization methods. SGD for strongly convex functions, which includes the bias-regularized SVM, has a sample complexity $O(\frac{1}{\lambda\varepsilon})$. Consequently, in the streaming setting we can simply maintain $O(\frac{1}{\lambda\varepsilon})$ random elements from the stream, then at the end

run SGD, yielding a space complexity of $O(\frac{d}{\lambda\varepsilon})$. Moreover, this is tight for general Lipschitz, strongly convex functions; there is an $\Omega(\frac{1}{\varepsilon})$ SGD sample complexity lower bound for this function class [1].

Some stochastic optimization methods like Stochastic Average Gradient (SAG) [10] achieve *linear* convergence, meaning that after T iterations $F_\lambda(\theta_T) - F_\lambda(\theta^*) \leq O(\rho^T)$ for some $\rho < 1$. However, in this case, $\rho \approx 1 - 1/n$, so we would need $T > n$ iterations, which is worse than $\frac{1}{\lambda\varepsilon}$ when n is sufficiently large.

Similarly, some stochastic optimization methods like Katyusha [2] achieve a sample complexity that has a dependence on ε like $1/\sqrt{\varepsilon}$. However, the sample complexity for such methods also has a sample complexity that scales linearly with n, again making them worse than $\frac{1}{\lambda\varepsilon}$ for the regime we care about.

Finally, note that if the elements of the stream are drawn IID from some distribution, and the size of the stream, n, is sufficiently large, then we can simply run online gradient descent (OGD) and use $O(d)$ space. However, we focus on the general setting where we cannot make any distributional assumptions.

Core-set and streaming algorithms for SVMs. Tsang et al. focuses on trying to speed up SVM optimization via *core-sets* [13]: a subset of training points that are sufficient for approximately optimizing the objective corresponding to the full training set. They approximately solve a Minimum Enclosing Ball (MEB) problem that they show is equivalent to the SVM. In the language of our paper, it shows an algorithm for achieving a $\pm\varepsilon$ additive approximation for the SVM objective in the batch setting using space $O(\varepsilon^{-2})$. [9] adapts these ideas to the streaming setting by showing a simple one-pass approximate MEB algorithm. However, they only achieve a *constant* approximation, rather than one with a target error ε. To the best of our knowledge, we are the first to analyze streaming SVM algorithms with sub-constant approximation guarantees.

1.3 Techniques

Multiplicative approximation algorithm. First note that by considering the points labeled $+1$ and -1 separately, the point estimation problem that we consider reduces to the following: given a set of n points, the goal is to sketch them such that later, given a query hyperplane denoted by $H = (\theta, b)$, one can estimate the sum of distances of the points on one side of H to H. In one dimension, the points are positioned on the real line and the query is also a value b on the real line and the goal is to compute $\sum_i \max\{0, b - x_i\}$.

The idea behind the streaming algorithm is the following: we uniformly sample a $1/2^i$ fraction of the input points, for $i = 1, 2, \ldots, O(\log n)$, and for each of the $O(\log n)$ sampling rates, we store the smallest $\tilde{O}(\epsilon^{-2})$ points that we have seen. Now given a query point q, we want to estimate the sum of distances of points $p < q$ to q. If there are fewer than $\tilde{O}(\epsilon^{-2})$ input points p less than q, then we have stored all of them and can compute this sum exactly. Otherwise we can think of partitioning the input points p into geometric scales, in powers of 2, based on their distance to q. The main insight is if one of these scales contributes to the overall sum, it must have a number n' of points in it of roughly the same order as the total number of points p' even further away from q. This is because each such point p' contributes even more than any point p in this scale to the sum. Consequently, if we choose i for which $1/2^i \approx 1/(n'\epsilon^2)$, then there will be about $\tilde{\Theta}(1/\epsilon^2)$ survivors in the sampling that are at this distance scale, and we will have stored all of them. By separately estimating the contribution of each scale and adding them together, we obtain our overall estimate. We note that to achieve our overall $\tilde{O}(\epsilon^{-2})$ space bound we need to obtain as crude of an additive error as

possible for scales that do not contribute as much as other scales to the overall objective. We do this by separately first estimating the contribution of each scale up to a constant factor, and then refining it appropriately.

Additive approximation algorithms. We also show *additive* approximation algorithms in the low dimensional regimes. As our lower bound depends on the diameter of the points, we assume without loss of generality that our point set has diameter 1. For $d = 1$, the sketching algorithm groups the points into $2/\sqrt{\varepsilon}$ groups in a way that i) each group has diameter at most $\sqrt{\varepsilon}$, and ii) each group has at most $n\sqrt{\varepsilon}$ points in it. It is easy to verify that such a grouping results in an additive approximation of ε. To make the algorithm work in the streaming setting, we must maintain the groups as the points arrive. Note that the partitioning based on the diameter can be done in advance. However in order to also partition based on the number of points (and make sure each group gets at most $n\sqrt{\varepsilon}$ points in it), we create a binary tree for each original group (that has diameter $\sqrt{\varepsilon}$). Whenever a group reaches its maximum size of $n\sqrt{\varepsilon}$, we create two children corresponding to that group and further points arriving in the group will be assigned to one of the children. Note that it is important that we cannot partition the previous points into the two children as we already discarded them in the stream.

For $d = 2$, we maintain a quad tree on the points in $[-1, 1] \times [-1, 1]$. Whenever a cell gets too many points we further partition it into four until its side length becomes too small. We process every cell of the quad tree so that if the query (which is a line now) does not collide with the cell, we can exactly compute the sum of distances of the points in the cell to the line (using [5]). Ignoring the points in the cells that the line crosses, will result in an algorithm with space usage of $O(\varepsilon^{-1})$. To push it down to $O(\varepsilon^{-4/5})$, we randomly sample a point from each cell and for the crossing cells, we will use the single sampled point to estimate the average distances of the points in the cell to the query line.

Finally, for the lower bounds in low dimensions, we develop reductions from the INDEXING problem. We show how to consider $k = \Omega(\varepsilon^{-\frac{d+1}{d+3}})$ positions inside a d-dimensional sphere, where they correspond to a bit-string of length k by Alice. If her i-th bit in the string is 1, then she will put n/k actual points in the corresponding positions, otherwise she does not put any point there. We show that Bob can recover Alice's input using hyper-plane queries.

Lower bound in high dimensions. We let $b = 0$ so that $F(\theta, 0) = \frac{1}{n}\sum_i \max\{0, \theta \cdot x_i\}$. This is similar to the subspace sketch problem studied in [7], which considers approximating $\sum_i \phi(\theta \cdot x_i)$ for $\phi(t) = |t|$ up to a multiplicative factor of $(1+\varepsilon)$. Here we have $\phi(t) = \max\{0, t\}$ instead (by flipping θ) and an additive error εn. The proof in [7] turns the multiplicative error into an additive error and so we can adapt the same approach in our current case. Following the same approach, we can show an $\Omega(d/(\varepsilon^2 \operatorname{polylog}(1/\varepsilon)))$ lower bound when $d = \Omega(\log(1/\varepsilon))$ for the point estimation problem. Below we sketch the main idea for the proof, which is similar to that in [7].

We show an $\tilde{\Omega}(1/\varepsilon^2)$ lower bound for $d = \Theta(\log(1/\varepsilon))$. The lower bound for general d follows from the concatenation of $\Theta(d/\log(1/\varepsilon))$ independent smaller hard instances.

In the remainder of this subsection let $d = \Theta(\log(1/\varepsilon))$ be such that $n = 2^d = \tilde{\Theta}(1/\varepsilon)$. Consider all the $\{-1, 1\}^d$ vectors and let x_i be the i-th $\{-1, 1\}$-vector scaled by some scalar r_i. Define a matrix $M \in \mathbb{R}^{n \times n}$, indexed by $\{-1, 1\}$-vectors, as $M_{ij} = \phi(\langle i, j\rangle)$. Then we have for $\theta \in \{-1, 1\}^d$ that $\sum_i \phi(-\langle \theta, x_i\rangle) = \sum_i \phi(\langle \theta, x_i\rangle) = \sum_i \phi(r_i\langle \theta, i\rangle) = \sum_i r_i\phi(\langle \theta, i\rangle) = \sum_i M_{\theta,i}r_i = \langle M_\theta, r\rangle$ if all $r_i \geq 0$, where M_θ denotes the θ-th row of M. Our goal is to encode random bits s_i in the scalars r_i, such that obtaining $\langle M, \theta\rangle$ within additive εn allows us to recover as many bits s_i as possible.

First we allow r_i to be negative and consider $(Mr)_\theta$. Let $r = \sum_i s_i \cdot \frac{M_i}{\|M_i\|_2}$, where $s_1, \ldots, s_n$ are i.i.d. Rademacher random variables. It follows from a standard concentration inequality that $\|r\|_\infty \leq \text{poly}(d)$. If M had orthogonal rows, then $\langle M_\theta, r\rangle = s_\theta \|M_\theta\|_2$, in which case we can recover the bit s_θ from the sign of $\langle M_\theta, r\rangle$, provided that $\|M_\theta\|_2$ is larger than the additive error εn.

However, M does not have orthogonal rows. The argument above still goes through so long as we can identify a subset $\mathcal{R} \subseteq [n] = [2^d]$ of size $|\mathcal{R}| = \Omega(2^d/\text{poly}(d))$ such that the rows $\{M_i\}_{i\in\mathcal{R}}$ are nearly orthogonal, meaning that the ℓ_2 norm of the orthogonal projection of M_i onto the subspace spanned by other rows $\{M_j\}_{j\in\mathcal{R}\setminus\{i\}}$ is much smaller than $\|M_i\|_2$. To this end, we study the spectrum of M using Fourier analysis on the hypercube, which shows that the eigenvectors of M are the rows of the normalized Hadamard matrix, while the eigenvalues of M are the Fourier coefficients associated with the function $g(s) = \phi(d - 2w_H(s))$, where $w_H(s)$ is the Hamming weight of a vector $s \in \{0,1\}^d$. It can be shown that M has at least $\Omega(2^d/\text{poly}(d))$ eigenvalues of magnitude $\Omega(2^{d/2}/\text{poly}(d))$. For the θ's which correspond to those eigenvalues, we have $\|M_\theta\|_2 = \Omega(2^{d/2}/\text{poly}(d))$ so that $\|M_\theta\|_2 = \Omega(\varepsilon n)$ for our choice of n and d, as required by the argument.

Recall that we require $r_i \geq 0$. Since $\|r\|_\infty \leq \text{poly}(d)$ with high probability, we can just shift each coordinate of r by a fixed amount of $\text{poly}(d)$ to ensure that all entries of r are positive. We can still obtain $\langle M_\theta, r\rangle$ with an additive error $O(\varepsilon 2^d \text{poly}(d))$, since the amount of the shift is fixed and bounded by $\text{poly}(d)$.

Last, rescaling θ and x_i's to unit vectors loses $\text{poly}(d) = \text{poly}(\log(1/\varepsilon))$ factors in the lower bound and we continue to have an $\Omega(1/(\varepsilon^2 \,\text{polylog}(1/\varepsilon)))$ lower bound.

Optimization lower bound. We prove optimization lower bounds by reducing from the INDEXING problem: Alice encodes points at specific locations on the unit sphere, then Bob uses the optimization sketch to decode whether a point was added at some particular location. The challenge is that Bob must be able to reason about a *single* data point when given access to an approximate optimum corresponding to an *entire* dataset. The key idea in getting this to work is for Bob to add some additional points with the following property: if the location being queried does not contain a point, then the added points are the *only* support vectors, i.e. (θ^*, b^*) is entirely determined by the added points, whereas if the the location being queried does contain a point, then (θ^*, b^*) is entirely determined by the added points and the point at that location. Hence, (θ^*, b^*) can take on exactly two possible values, and does not depend on any of the remaining points in the dataset. Moreover, the strong convexity of F_λ allows us to upper bound $\|(\theta^*, b^*) - (\hat{\theta}, \hat{b})\|$ in terms of ε. We make this precise in the following lemma, which we will refer to multiple times later on:

▶ **Lemma 3.** *If $F_\lambda(\hat{\theta}, \hat{b}) \leq F_\lambda(\theta^*, b^*) + \varepsilon$, then $\|(\hat{\theta}, \hat{b}) - (\theta^*, b^*)\|_2 \leq \sqrt{2\varepsilon/\lambda}$.*

By exactly characterizing what (θ^*, b^*) is in these two possible cases, and showing that the gap is more than $2\sqrt{2\varepsilon/\lambda}$, we show that Bob can distinguish these two situations and decode the bit. The analysis is delicate, requiring a carefully chosen construction for the proof to go through.

Open Problems. For the case of $d = 1$, we have matching upper and lower bounds for (additive) point estimation of $\Theta(\varepsilon^{-1/2})$. This also translates into an *optimization* upper bound of $O(\varepsilon^{-1/2})$, the best upper bound we know of for this setting. However, we have no optimization lower bound for $d = 1$. Instead, we have an optimization lower bound of $\Omega(\varepsilon^{-1/4})$ for $d = 2$ and $\Omega(\varepsilon^{-1/2})$ for $d \geq 3$. Moreover, for optimization in high dimensions, there remains a gap between the $\Omega(\varepsilon^{-1/2})$ lower bound and $O(1/\varepsilon)$ upper bound. It remains to close all of these gaps.

Also, while in this work we focus on linear SVMs, often times non-linear kernels are preferred in practice. This raises the question of whether we can extend our results to this setting. One approach is to use random feature maps that allow one to convert a kernel SVM problem into a linear SVM problem [8]. However, this increases the dimension significantly, so that sampling and running SGD is more efficient than optimization via point estimation.[2]

2 Point Estimation

In this section, we study the streaming complexity of the point estimation problem. Specifically, the algorithm sees the data points (x_i, y_i), for $1 \le i \le n$, one by one. The goal is to keep a sketch of the data such that later, given the query parameters (θ, b), it can output an estimate of the SVM objective function $F(\theta, b)$.

Setup for point estimation. We can simplify the presentation by focusing on a slight simplification of the SVM objective (without loss of generality). First, we note that, since $y_i \in \{+1, -1\}$, we can estimate the contribution from x_i's with $y_i = +1$ and $y_i = -1$ separately. Hence, for point estimation it is enough to assume that $y_i = +1$, as well as that $\lambda = 0$ (since the regularization can be computed independently of the data). Furthermore, we can just work with the following related objective:

$$F(\theta, b) := \frac{1}{n} \sum_{i=1}^{n} \max\{0, b - \theta \cdot x_i\}, \tag{3}$$

by adjusting b accordingly. Hence we focus on estimating the function from Eqn. (3) for the rest of this section.

Note that when $d = 1$, it is enough to consider the case of $\theta = +1$. First, because for $\theta \in (0, 1)$, we can rescale the output of a sketch (that uses $\theta = +1$ and accordingly rescaled b). Second, because the case of $\theta \in [-1, 0)$ is precisely symmetric, so one can just keep two sketches, one for each of $\theta \in \{-1, +1\}$. Note that the objective simplifies to $F = \frac{1}{n} \sum_{i=1}^{n} \max\{0, q - x_i\}$ where $q = b$. We will call the value q the query.

2.1 Multiplicative $(1 + \varepsilon)$ approximation algorithm for $d = 1$

Here we consider the case of $d = 1$: We are given a set of n points $x_i \in \mathbb{R}$, and given any query $q \in \mathbb{R}$, the goal is to approximate $\sum_{i: x_i \le q} (q - x_i)$ up to a multiplicative $1 + \varepsilon$ factor. To analyze the bit complexity of the problem, we assume the points are integers between 1 and W. A simple sketching algorithm is given in the full version. Here we present a streaming algorithm for the problem.

Streaming. Here we assume that the values $x_1, \ldots, x_n$ are given in a stream in this order, and we are allowed to make a single pass over it, and the query q is given at the end of the stream. Note that x_i's are not necessarily sorted, and for simplicity, we assume all x_i's are distinct. The algorithm maintains the following sketch throughout the stream.

[2] As described in [12], one can use adapt SGD to work for kernelized SVMs. This involves tracking dual variables α_i, which we can do with the same space complexity of $O(\frac{d}{\lambda \varepsilon})$ as before.

Sketch. The sketch consists of two collections of sets of samples as described below. The first collection is used to get a crude (constant factor) approximation of the contribution of each contributing interval as defined later, and the second collection is used for a more precise approximation.

- For each $0 \leq i \leq \log n$, sample every point with probability $1/2^i$, and preserve the $m_1 = C_1\varepsilon^{-1}\log^2 W$ points with the smallest x value in the set E_i, where C_1 is a constant to be specified later.
- For each $0 \leq i \leq \log n$, sample every point with probability $1/2^i$, and preserve the $m_2 = C_2\varepsilon^{-2}\log W$ points with the smallest x value in the set S_i, where C_2 is a constant to be specified later.

▶ **Observation 4** (Space). *The sets E_i and S_i can be maintained in a stream. Let $M = \max\{m_1, m_2\}$, then the space usage of the algorithm is $O(\log n \cdot M \cdot \log W) = O(\frac{\log n \cdot \log^2 W}{\varepsilon}(\frac{1}{\varepsilon} + \log W))$ bits.*

Next we describe the query processing algorithm. The analysis is deferred to Appendix A.

Query algorithm. Let p be the largest value in $S_0 \cup E_0$. Given the query point $q \in \mathbb{R}$, we proceed as follows.

- If $q \leq p$, then we can report an exact solution using the corresponding sample set: e.g. if $p \in S_0$, then we output $\sum_{x \in S_0, x \leq q}(q - x)$.
- Otherwise, we group the points based on their distance to q and estimate the contribution of each group separately. More precisely, let $D = q - p$, which is a positive number, and for each $1 \leq j \leq \log D$, define the interval $R_j = (q - \frac{D}{2^{j-1}}, q - \frac{D}{2^j}]$. For notational convenience, let R_0 be the interval covering $S_0 \cap E_0$ which ends at the point p. Finally for each $0 \leq j \leq \log D$, let $t_j = |P \cap R_j|$ be the number of points in the interval R_j, and $T_j = |P \cap (\bigcup_{k<j} R_k)|$ be the number of points to the left of interval R_j.
- Let $i'(j)$ be the largest i such that E_i contains at least $\log D$ points from R_j. If no such i' exists, let $i'(j) = -1$. The value $i'(j)$ shows which sampled set ($E_{i'(j)}$) we should use for our crude approximation. As we show in Lemma 17, if $i'(j) = -1$, the contribution of the points in R_j can be ignored).
- Let $\phi_j = \min\{1, \frac{|E_{i'(j)} \cap R_j|}{|E_{i'(j)} \cap (\cup_{k<j} R_k)|}\}$. This value is used to approximate the ratio of the points in R_j to the points that are to the left of R_j, i.e., $\phi_j \approx \frac{t_j}{T_j}$. This is verified in Lemma 18.
- We set the value of $i(j)$ as follows.
  - If $i'(j) = -1$ or $\phi_j \leq \frac{\varepsilon}{\log W}$, then set $i(j) = -1$. In this case, the contribution of R_j can be ignored.
  - Otherwise, if $\phi_j \geq \frac{1}{\log W}$, then set $i(j)$ to be the largest i such that S_i contains at least $1/\varepsilon^2$ points from R_j. If no such i exists let $i(j) = -1$. This case in analyzed in Lemma 19 and Lemma 20.
  - Finally, if $\frac{\varepsilon}{\log W} \leq \phi_j \leq \frac{1}{\log W}$, then set $i(j)$ to be the largest i such that S_i contains at least $(\phi_j \log W/\varepsilon)^2$ points from R_j. This case in analyzed in Lemma 21 and Lemma 22.
- Report $\sum_{j \leq \log D, i(j) \neq -1} \sum_{x \in S_{i(j)} \cap R_j} 2^i(q - x)$. That is for all sets whose contribution is significant (equivalently $i(j) \neq -1$) we estimate their contribution using sample set $S_{i(j)}$.

We then have the following results.

▶ **Lemma 5** (main lemma). *This algorithm returns a $(1 + O(\varepsilon))$ multiplicative approximation.*

▶ **Corollary 6.** *There exists a one pass streaming algorithm that computes a $(1+\varepsilon)$ multiplicative approximation for point estimation variant of the problem in one dimensional case. Moreover, if the points come from $[W]$, the space usage of the algorithm is $O(\frac{\log^2 n \cdot \log W}{\varepsilon}(\log n + 1/\varepsilon))$ bits.*

2.2 Lower Bounds

We now show that one cannot hope to get a sketch with multiplicative approximation in higher dimensions than one with a bound independent of n. In fact we show the following additive approximation lower bound for any sketching algorithm (and hence streaming algorithm as well).

▶ **Theorem 7.** *For any $d \geq 1$, $\varepsilon \in (0,1)$, and $n \geq 1/\varepsilon$, there exists an instance of the problem, where the point set has diameter $O(1)$, such that getting an algorithm with additive approximation factor ε, requires space of $\Omega(\varepsilon^{-(d+1)/(d+3)})$ bits.*

Moreover, getting a $(1+\varepsilon)$-multiplicative approximation for $d \geq 2$ requires $\Omega(n)$ space.

The proof is given in Appendix B. We note that while this theorem formally applies to the simplified objective, it immediately translates to the SVM objective as well as we only use points with one label (when the problems are exactly equivalent).

The preceding theorem gives at most an $\Omega(1/\varepsilon)$ lower bound, leaving a quadratic gap from the simple random sampling algorithm of $\tilde{O}(d/\varepsilon^2)$ bits. In fact, for high dimensions $d = \Omega(\log(1/\varepsilon))$, we can prove a lower bound of $\tilde{\Omega}(d/(\varepsilon^2 \operatorname{polylog}(1/\varepsilon))$ bits, tight up to logarithmic factors. We state the theorem below and give the proof in the full version.

▶ **Theorem 8** (high-dimensional)**.** *There exist constants $C \in (0,1]$ and $\varepsilon_0 > 0$ such that the following holds. Let $d_0 = 2\log_2(C/(\varepsilon \operatorname{polylog}(1/\varepsilon))$. For any $\varepsilon \in (0, \varepsilon_0)$, $d \geq d_0$, $n \geq (d/d_0)2^{d_0}$, any algorithm that gives an additive ε-approximation for the point estimation problem with probability at least $2/3$ requires $\Omega\left(d/(\varepsilon^2 \operatorname{polylog}(1/\varepsilon))\right)$ bits.*

2.3 Additive approximation algorithms

We now design streaming algorithms that achieve an additive ε-approximation to the objective Eqn. (3). We also generalize these results to a slightly modified objective in the full version. We start with dimension $d = 1$.

▶ **Theorem 9.** *There exists a one pass streaming algorithm for the point estimation variant of the problem in the one dimensional regime, that achieves an additive error of ε, space of $O(\varepsilon^{-1/2}\sqrt{\log(1/\varepsilon)})$ words, and that succeeds with constant probability per query.*

Recall that for $d = 1$, the objective simplifies to $F(q) = \frac{1}{n}\sum_{i=1}^{n} \max\{0, q - x_i\}$. We describe a *sketching* algorithm that produces a sketch of size $O(1/\sqrt{\varepsilon})$ that is able to answer point estimation queries to this F. Later, we show how to adapt this algorithm to the streaming setting.

Let $m = (1/\sqrt{\varepsilon})$ and consider two sets of m points. First consider $Y_1, \ldots, Y_m$ such that Y_i is at position i/m. Moreover consider m points $X_1, \ldots, X_m$ such that X_i is at position $x_{(i \cdot n)/m}$, where we assume that x_i's are in a sorted order, i.e., $x_1 \leq \cdots \leq x_n$. Now sort these $2m$ points and name them $Z_1, \ldots, Z_{2m}$. For each $i \leq 2m$ we store three numbers: i) Z_i itself, ii) s_i, the sum of the distances of the points to the left of Z_i to the point Z_i, and iii) c_i, the number of points x_i to the left of Z_i.

Given a query $q \in [0,1]$, we will find i such that $Z_i \leq q < Z_{i+1}$. We will return $s_i + c_i \cdot (q - Z_i)$. Clearly for the the points that are to the left of Z_i this distance is computed correctly. The only points that are not computed in the sum are part of the points in the

interval $[Z_i, Z_{i+1}]$, but we know that there are at most $n\sqrt{\varepsilon}$ of them (by our choice of the X_i's) and their distance to q is at most $(Z_{i+1} - Z_i) \leq \sqrt{\varepsilon}$ (by our choice of Y_i's). Therefore they introduce an average error of at most ε as we require.

Streaming. We adapt the above algorithm to the streaming setting as follows. We keep a binary tree, where each node corresponds to an interval in $[-1, 1]$ (the domain of x_i). The root corresponds to the entire interval $[-1, 1]$, and the two children of a node/inverval are the 2 half correspondingly (applied recursively). Initially, we start with a tree of height $\log_2 1/\sqrt{\varepsilon}$, where the leaves correspond to intervals of length precisely $\sqrt{\varepsilon}$ (assuming it's a power of two, w.l.o.g.).

As we stream through the points x_i, we add the information about the point x_i to the leaf corresponding to the interval containing x_i. In particular, each node v, with associated interval I_v, keeps a count of points c_v, as well as s_v which is the sum, over of the points accounted in c_v, of their distance to the right border of the interval.

We may also expand this leaf v to add its two children (v ceases to be a leaf). The leaf is split when c_v reaches value $\sqrt{\varepsilon}n$. The new children start with their counters equal to 0. One exception is that if the depth of the node is more than $3\log(1/\varepsilon)$ (the interval's diameter is $< \varepsilon$), in which case we don't do the expansion.

To answer a query $q \in [-1, 1]$, we sum up, over all nodes v (internal nodes and leaves) whose interval I_v is entirely to the left of q, the quantity $s_v + c_v \cdot (q - I_v)$, where $q - I_v$ is the distance from q to (the rightmost endpoint of) I_v. The analysis is given in Appendix C.1.

Now we study dimension $d = 2$. We now develop a streaming algorithm for sketching a set of points in the 2D plane such that given any query (affine) line in the plane, one can approximate the cost. To simplify the ensuing notation, we denote the set of points by $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n) \in [0, 1] \times [0, 1]$, and the query by a line $\{x : \theta^T x = b\}$, which we denote by $L = (\theta, b)$. Recall the assumption that $\|\theta\| \leq 1$, we may assume that $\|\theta\| = 1$. Our goal is equivalent to reporting the sum of distances of the points on one side of L to L. Henceforth we denote the distance from point p to line L as $D(p, L)$.

▶ **Theorem 10.** *There is a streaming algorithm for the point estimation problem in two dimensions, that with constant probability, achieves additive error ε, with sketch size $O(\varepsilon^{-4/5})$ words.*

The following shows how to get an $O(1/\varepsilon)$-size sketch with an additive error of $O(\varepsilon^{5/4})$, and at the end we just replace $\varepsilon' = \Theta(\varepsilon^{4/5})$ to prove the above theorem.

We use a quad-tree over the unit square $[0, 1] \times [0, 1]$, where each node is associated with a number of points (each point is associated with exactly one node of the quad-tree). Thus each node v contains a counter c_v for the number of associated points, a randomly chosen associated points (chosen using reservoir sampling), as well as a sketch S_v to be described later. Initially, the quad-tree is of depth $\log(1/\sqrt{\varepsilon})$ and all counters/sketches are initialized to zero. When we stream over a point p_i, we associate it with the corresponding leaf of the quad-tree (process defined later), unless the counter c_v is already $\varepsilon \cdot n$ and the depth is at least $2\log(1/\varepsilon)$ (i.e., the side length is at least ε^2). In that case the leaf v is expanded by adding its 4 children, which become new leaves (with counters initialized to 0).

When we associate a point with a node v, we increment c_v and update the sketch S_v on the associated points. The sketch S_v for the associated points, say termed P_v, allows us to compute, for any query line $L = (\theta, b)$, the sum $\sum_{p \in P_v} (b - \theta^T p)$. The sum $\sum_{p \in P_v} \theta^T p$ can be computed in a streaming fashion using the sketch from [5]. In particular, the sketch actually consists of two counters: X_v, the sum of the x coordinates, and Y_v, the sum of the y coordinates.

Query algorithm. Given a query line L, we distinguish contribution from points in two types of quad-tree nodes: nodes that do not intersect the line and those that do. For the first kind, we can just use the sketch S_v to estimate the distance to the line, without incurring any error. More precisely we have that $\sum_{p\in P_v} D(p,L) = \sum_{p\in P_v}(b - \langle p,\theta\rangle) = c_v b - \langle (X_v, Y_v),\theta\rangle$. Note that this is included in the final sum iff the entire node lies in the halfplane $\langle x,\theta\rangle \le b$.

For the second kind of nodes, we estimate their contribution as follows. For each non-empty node v, with the random sample r_v, we add to the final sum the quantity $(1/n)\cdot c_v \cdot \max\{0, b - \theta^T r_v\}$.

The analysis is deferred to Appendix C.2.

3 Optimization

In this section we consider the problem of finding the (approximate) optima for the SVM objective in the streaming setting. First, we show that a streaming solution for the point estimation problem immediately leads to a solution for the optimization problem, with only a $O(d\log 1/\varepsilon)$ loss in space complexity. Second, we give lower bounds for the optimization problem, showing a (different) polynomial dependence on $1/\varepsilon$ is still required for dimension $d > 1$.

As before, we consider the SVM optimization problem in which the bias is regularized: $\min_{\theta,b} F_\lambda(\theta, b)$, where F_λ is as defined in (1). Without loss of generality, we assume that the inputs are contained in a ball of radius 1, i.e., $\|x_i\| \le 1$, and that $y_i \in \{-1, +1\}$.

Recall that [12] show that $O(1/(\lambda\varepsilon))$ random samples (x_i, y_i) are enough for computing an ε-approximate optimum (by running SGD). This can be seen as a streaming algorithm with space complexity $O(d/(\lambda\varepsilon))$. We show that, given an efficient streaming algorithm for point estimation, we can solve the optimization problem with only a minor blowup.

▶ **Theorem 11.** *Suppose there is a streaming algorithm that, after seeing data $\{(x_i, y_i)\}_{i=1}^n$, where $\|x_i\| \le 1$, can produce a sketch of size s that, given any (θ, b) such that $\|(\theta, b)\| \le \sqrt{2/\lambda}$, is able to output $\hat{F}(\theta, b)$ such that $|\hat{F}(\theta, b) - F(\theta, b)| \le \varepsilon$ with probability at least 0.9. Then there is also a streaming algorithm that, under the same input, will output $(\hat{\theta}, \hat{b})$ with $|F_\lambda(\hat{\theta}, \hat{b}) - F_\lambda(\theta^*_\lambda, b^*_\lambda)| \le \varepsilon$ with probability at least 0.9, while using space $O(s \cdot d\log d/(\lambda\varepsilon))$.*

The proof of this theorem is a standard net argument and is given in the full version.

Recall that our point estimation results assume $\|(\theta, b)\| \le 1$, which can be adapted to $\|(\theta, b)\| \le R$ by replacing ϵ with ϵ/R. Letting $R = \sqrt{2/\lambda}$, the above theorem implies that we get an optimization algorithm for $d = 1$ that uses $\tilde{O}(\varepsilon^{-1/2}\lambda^{-1/4})$ space, and an optimization algorithm for $d = 2$ that uses $\tilde{O}(\varepsilon^{-4/5}\lambda^{-2/5})$ space. Note that this has a polynomially better dependence on both ε and λ relative to the $O(1/\lambda\varepsilon)$ bound that we get from SGD.

Lower bounds. We start with the high-dimensional case. Suppose there exists a sketch such that, given a stream of inputs $\{(x_i, y_i)\}_{i=1}^n$, $\|x_i\|_2 \le 1$, with probability at least 0.9 outputs some $(\hat{\theta}, \hat{b})$ such that $F_\lambda(\hat{\theta}, \hat{b}) \le F_\lambda(\theta^*, b^*) + \varepsilon$, where $F_\lambda(\theta, b) := \frac{\lambda}{2}(\|\theta\|^2 + b^2) + \frac{1}{n}\sum_{i=1}^n \max\{0, 1 - y_i(\theta^T x_i + b)\}$. For now, suppose $\lambda = 10^{-4}$ and $d = O(\log n)$. Later, we show a similar lower bound for low dimensions when $\lambda = \Theta(1/n)$.

▶ **Theorem 12.** *Such a sketch requires $\Omega(\varepsilon^{-1/2})$ words of space.*

Proof idea: We will reduce from the INDEXING problem in the one-way communication model. At a high level, we will argue that we can query whether a point exists at a given location on the unit sphere by adding additional points with the property that the optimal

parameters are determined entirely by the added points and the point being queried (if it exists). This yields a separation in the optimal parameters in these two cases, which we will argue (via strong convexity) is distinguishable using such a sketch.

Specifically, suppose Alice is given a bit string $b \in \{0,1\}^{n/2}$ which she wants to encode. Let $T = \{v_1, \ldots, v_n\}$ be a subset of the unit sphere in d dimensions satisfying $\forall v_1 \neq v_2 \in T$, $v_i^T v_j < 1 - 10\delta$, where δ will be specified later, with $|T| = n$. For $\delta = \frac{1}{100}$, such a set exists for $d = \text{poly}(\log n)$. If $b_i = 0$, Alice adds $(v_i, -1)$ to the sketch S; otherwise, if $b_i = 1$, Alice adds $(v_{n/2+i}, -1)$ to S. Alice then sends the sketch to Bob, who decodes b_q for $q \in [\frac{n}{2}]$ by querying whether a point exists at location x_q. In particular, Bob adds $\frac{n}{4}$ copies of $(x_\alpha, -1)$, and $\frac{n}{4}$ copies of $(x_\beta, +1)$, where $x_\alpha := (1-\delta)x_q$ and $x_\beta := (1+\delta)x_q$. Let $(\hat{\theta}, \hat{b})$ be the output of the sketch after doing so. Define $\theta_0 := \frac{2\lambda(1+\delta)+\delta}{2\lambda(1+(1+\delta)^2)}$ and $\theta_1 := \frac{2\lambda(1+\delta)+\delta(1+\frac{1}{n})}{2\lambda(1+(1+\delta)^2)}$. If $\|\hat{\theta} - \theta_0\| < \|\hat{\theta} - \theta_1\|$ then Bob outputs "$b_i = 0$"; otherwise, Bob outputs "$b_i = 1$".

3.1 Preliminaries for the lower bound

We introduce some standard facts about the SVM objective. We can rewrite the SVM objective as a constrained optimization problem:

$$\min_{\theta,b} \frac{\lambda}{2}\|(\theta,b)\|_2^2 + \frac{1}{n}\sum_{i=1}^{n}\gamma_i \tag{4}$$

subject to $\gamma_i \geq 0$ and $\gamma_i \geq 1 - y_i(\theta^T x_i + b)$.

The corresponding Lagrangian is then:

$$\min_{\theta,b,\gamma,\eta,\alpha} \mathcal{L} := \frac{\lambda}{2}\|(\theta,b)\|_2^2 + \frac{1}{n}\sum_{i=1}^{n}\gamma_i - \sum_{i=1}^{n}\gamma_i\eta_i + \sum_{i=1}^{n}\nu_i[1 - \gamma_i - y_i(\theta^T x_i + b)] \tag{5}$$

subject to: $\gamma_i \geq 0$, $\gamma_i \geq 1 - y_i(\theta^T x_i + b)$, $\eta_i \geq 0$, and $\nu_i \geq 0$.

The KKT conditions are:

$$\eta_i\gamma_i = 0 \tag{6}$$
$$\nu_i(1 - \gamma_i - y_i(\theta^T x_i + b)) = 0 \tag{7}$$
$$\theta^* = \frac{1}{\lambda}\sum_{i=1}^{n}\nu_i y_i x_i \tag{8}$$
$$b^* = \frac{1}{\lambda}\sum_{i=1}^{n}\nu_i y_i \tag{9}$$
$$\eta_i = \frac{1}{n} - \nu_i \tag{10}$$
$$\gamma_i = \max\{0, 1 - y_i(\theta^T x_i + b)\} \tag{11}$$
$$\eta_i \geq 0, \nu_i \geq 0 \tag{12}$$

3.1.1 Notation

We will characterize the optimal solutions for the two scenarios (there exists a point at x_q or there does not) for the case that the dimension is $d = 1$. We will later show how this provides results for $d > 1$. We will specify δ later, but will always maintain the relation $\lambda = \delta^2$.

Let $n = \frac{1}{20\sqrt{\varepsilon}}$, and define a set S_0 of n points as follows:

- $\frac{n}{4}$ points are $x_\alpha := 1 - \delta$ with $y_\alpha := -1$.
- $\frac{n}{4}$ points are $x_\beta := 1 + \delta$ with $y_\beta := +1$.
- The remaining $\frac{n}{2}$ points are not support vectors and are otherwise arbitrary; i.e. are such that $1 - y((\theta_0^*)^T x + b_0^*) < 0$, where

$$(\theta_0^*, b_0^*) = \arg\min_{(\theta,b)} F_\lambda^{(0)}(\theta, b) := \frac{\lambda}{2}(\|\theta\|^2 + b^2) + \frac{1}{n}\sum_{(x,y)\in S_0} \max\{0, 1 - y(\theta^T x + b)\} \quad (13)$$

Observe that such points exist as long as $\theta_0^* \neq 0$, which is easy to show.

Define $\gamma_\alpha^{(0)} := \max\{0, 1 - y_\alpha((\theta_0^*)^T x_\alpha + b_0^*)\} = \max\{0, 1 + \theta_0^*(1-\delta) + b_0^*\}$ and $\gamma_\beta^{(0)} := \max\{0, 1 - y_\beta((\theta_0^*)^T x_\beta + b_0^*)\} = \max\{0, 1 - \theta_0^*(1+\delta) - b_0^*\}$.

Similarly, define S_1 to be the set of n points that is exactly the same as S_0, except that instead of $\frac{n}{2}$ points that are not support vectors, there are $\frac{n}{2} - 1$ points that are not support vectors and one additional point at $x_q := 1$ with $y_q := -1$. Similarly define $F_\lambda^{(1)}$, (θ_1^*, b_1^*), $\gamma_\alpha^{(1)}$, and $\gamma_\beta^{(1)}$ to be the analogous quantities where S_0 is replaced with S_1, and let $\gamma_q^{(1)} := \max\{0, 1 - y_\beta((\theta_1^*)^T x_q + b_1^*)\} = \max\{0, 1 + \theta_1^* + b_1^*\}$. We will also sometimes use the notation $\gamma_\alpha(\theta, b) := \max\{0, 1 - y_\alpha(\theta^T x_\alpha + b)\}$, and similarly for $\gamma_\beta(\theta, b)$.

3.1.2 Lemmas

Next, we show some properties of the optimal solutions in each of the two cases.

▶ **Lemma 13.** *Suppose $\delta < \frac{1}{100}$ and $n \geq \frac{1}{\delta^2}$. Then we have that $\gamma_\alpha^{(i)} > 0$ and $\gamma_\beta^{(i)} = 0$ for $i \in \{0, 1\}$.*

Proof. To see that $\gamma_\alpha^{(i)} > 0$ and $\gamma_\beta^{(i)} = 0$ for $i \in \{0, 1\}$, we will rule out the other possibilities.

▷ Claim. Suppose $\delta < \frac{1}{7}$ and $n \geq \frac{1}{\delta^2}$. Then we cannot have $\gamma_\beta^{(i)} > 0$.

Suppose, for the sake of contradiction, that $\gamma_\beta^{(i)} > 0$, and let (θ_i^*, b_i^*) be the corresponding optimal parameters. First consider $i = 0$. Then since $\gamma_\beta^{(0)} > 0$, by the KKT conditions we have that the dual variables corresponding to each of the $\frac{n}{4}$ copies of x_β, call them ν_β, satisfy $\nu_\beta = \frac{1}{n}$. Moreover, note that this is the maximal value possible for the dual variables. Additionally, for points that are not support vectors, the dual variables are $\nu_i = 0$. The KKT conditions then imply:

$$\theta_0^* = \frac{1}{\lambda}\sum_{i=1}^n \nu_i x_i y_i \geq \frac{1}{\lambda}\left(\frac{n}{4}\frac{1}{n}(1+\delta) - \frac{n}{4}\frac{1}{n}(1-\delta)\right) = \frac{\delta}{2\lambda} = \frac{1}{2\delta} \quad (14)$$

and $b_0^* = \frac{1}{\lambda}\sum_{i=1}^n \nu_i y_i \geq 0$. But this implies that $1 - \theta_0^*(1+\delta) - b_0^* \leq 1 - \frac{1}{2\delta}(1+\delta) = \frac{1}{2}(1 - \frac{1}{\delta}) < 0$ (since $\delta < 1$), which contradicts that $\gamma_\beta^{(0)} > 0$.

Next, consider when $i = 1$. This time, we also have x_q as a support vector. Again using the fact that the corresponding dual variable is at most $\nu_q \leq \frac{1}{n}$, we have:

$$\theta_1^* = \frac{1}{\lambda}\sum_{i=1}^n \nu_i x_i y_i \geq \frac{1}{\lambda}\left(\frac{\delta}{2} - \frac{1}{n}\right) \geq \frac{1}{\lambda}\left(\frac{\delta}{2} - \delta^2\right) \geq \frac{1}{\lambda}\left(\frac{\delta}{2} - \frac{\delta}{4}\right) = \frac{1}{4\delta} \quad (15)$$

and $b_1^* = \frac{1}{\lambda}\sum_{i=1}^n \nu_i y_i \geq -\frac{1}{\lambda n}$. But this implies that $1 - \theta_1^*(1+\delta) - b_1^* \leq 1 - \frac{1}{4\delta}(1+\delta) + \frac{1}{\delta^2 n} \leq \frac{1}{4}(3 - \frac{1}{\delta}) + 1 < 0$, contradicting that $\gamma_\alpha^{(1)} > 0$ and proving the claim.

▷ Claim. It cannot hold that $\gamma_\alpha^{(i)} = \gamma_\beta^{(i)} = 0$ for either $i \in \{0, 1\}$.

First consider $i = 0$. We will show that in this case $\theta_0^* = \frac{1}{\delta}$ and $b_0^* = -\frac{1}{\delta}$. By assumption, we know that $1 + \theta^* + b^* - \theta^*\delta \leq 0$ and $1 - (\theta^* + b^*) - \theta^*\delta \leq 0$. Summing these implies that

$$1 - \theta_0^*\delta \leq 0 \Rightarrow \theta_0^* \geq \frac{1}{\delta} \tag{16}$$

Combining this last result with $\gamma_\alpha^{(0)} = 0$:

$$0 \geq 1 + \theta_0^*(1 - \delta) + b_0^* \geq 1 + \frac{1}{\delta}(1 - \delta) + b_0^* = \frac{1}{\delta} + b_0^* \Rightarrow b_0^* \leq -\frac{1}{\delta} \tag{17}$$

We have that $\theta = \frac{1}{\delta}$ and $b = -\frac{1}{\delta}$ satisfy $\gamma_\alpha(\theta, b) = \gamma_\beta(\theta, b) = 0$. By these last two equations, they are also clearly the *smallest norm* values satisfying these; hence they indeed minimize the overall optimization problem. But then $F_0(\theta, b) = \frac{\lambda}{2}(\frac{2}{\delta^2}) = 1$. But since we also have that $F_0(0, 0) = 1$, this cannot be the optimal solution, so we indeed cannot have $\gamma_\alpha^{(0)} = \gamma_\beta^{(0)} = 0$.

Now consider when $i = 1$. By the same argument as before, we must again have that if $\gamma_\alpha^{(1)} = \gamma_\beta^{(1)} = 0$ then $\theta_1^* \geq \frac{1}{\delta}$ and $b_1^* \leq -\frac{1}{\delta}$. Observe also that $F_1(\theta, b) = F_0(\theta, b) + \frac{1}{n}\gamma_q(\theta, b) \geq F_0(\theta, b)$. Hence:

$$F_1(\theta_1^*, b_1^*) \geq F_0(\theta_1^*, b_1^*) = \frac{\lambda}{2}(2\frac{1}{\delta^2}) = 1 \tag{18}$$

which again contradicts the optimality of (θ_1^*, b_1^*) since $F_1(0, 0) = 1$. The proof of the claim is complete.

Since under the assumption of the lemma we cannot have $\gamma_\alpha^{(i)} = \gamma_\beta^{(i)} = 0$, and we cannot have that $\gamma_\beta^{(i)} > 0$, we must have that $\gamma_\alpha^{(i)} > 0$ and $\gamma_\beta^{(i)} = 0$, concluding the proof. ◀

Next, we show what the optimal parameters are in this case.

▶ **Lemma 14.** *Suppose that $\gamma_\alpha^{(0)} > 0$ and $\gamma_\beta^{(0)} = 0$. Then we have:*

$$\theta_0^* = \frac{2\lambda(1 + \delta) + \delta}{2\lambda(1 + (1 + \delta)^2)} \tag{19}$$

and $b_0^* = 1 - (1 + \delta)\theta_0^*$.

Similarly, if $\gamma_\alpha^{(1)} > 0$ and $\gamma_\beta^{(1)} = 0$. Then we have:

$$\theta_1^* = \frac{2\lambda(1 + \delta) + \delta(1 + \frac{1}{n})}{2\lambda(1 + (1 + \delta)^2)} \tag{20}$$

and $b_1^* = 1 - (1 + \delta)\theta_1^*$.

Proof. First, observe that for $i \in \{0, 1\}$ we not only have $\gamma_\beta^{(i)} = \max\{0, 1 - (\theta_i^* + b_i^*) - \theta_i^*\delta\} = 0$, but also have that

$$1 - (\theta_i^* + b_i^*) - \theta_i^*\delta = 0 \tag{21}$$

To see this, suppose otherwise, that $1 - (\theta_i^* + b_i^*) - \theta_i^*\delta < 0$. Then there exists some $0 < \theta' < \theta_i^*$ such that we still have $1 - (\theta' + b_i^*) - \theta'\delta = 1 - \theta'(1 + \delta) - b_i^* < 0$. Moreover, this can only decrease the regularization term, $\frac{\lambda}{2}\theta^2$, and can only decrease the $\gamma_\alpha^{(i)} = \max\{0, 1 + \theta(1 - \delta) + b\}$ term. In the case that $i = 1$, this can also only decrease the $\gamma_q^{(1)} = \max\{0, 1 + \theta + b\}$ term. Either way, this contradicts the optimality of θ_i^*. Hence, $b_i^* = 1 - \theta_i^*(1 + \delta)$.

For $i = 0$, plugging this in and simplifying, we can rewrite the optimization problem as:

$$\min_{\theta} \frac{\lambda}{2}(\theta^2 + (1 - \theta(1+\delta))^2) + \frac{1}{2}(1 - \theta\delta) \tag{22}$$

Differentiating with respect to θ and setting the expression equal to zero,

$$\begin{aligned} 0 &= \frac{\partial}{\partial\theta}\left[\frac{\lambda}{2}(\theta^2 + (1-\theta(1+\delta))^2) + \frac{1}{2}(1-\theta\delta)\right]\Big|_{\theta=\theta_0^*} \\ &= \frac{\lambda}{2}(2\theta_0^* - 2(1-\theta_0^*(1+\delta))(1+\delta)) - \frac{\delta}{2} \\ &= \lambda\theta_0^*(1 + (1+\delta)^2) - \lambda(1+\delta) - \frac{\delta}{2} \end{aligned}$$

from which we can solve for θ_0^* and obtain that

$$\theta_0^* = \frac{2\lambda(1+\delta)+\delta}{2\lambda(1+(1+\delta)^2)}.$$

Similarly, for $i = 1$, we can rewrite the optimization problem as:

$$\min_{\theta} \frac{\lambda}{2}(\theta^2 + (1 - \theta(1+\delta))^2) + \frac{1}{4}(1 + \theta + 1 - \theta(1+\delta) - \theta\delta) + \frac{1}{n}(1 + \theta + 1 - \theta\delta) \tag{23}$$

We can again differentiate with respect to θ and set the expression equal to zero, then solve for θ_1^*, yielding the desired expression. ◀

We now adapt these results to the case that we care about: when $d > 1$ and there are points other than just x_α, x_β, and x_q.

First, note that when $d > 1$ and the only possible support vectors are x_α, x_β, and x_q, $\theta_i^* \in \mathbb{R}^d$ is parallel to x_q. Hence, θ_i^* reduces to the one-dimensional case, and is simply projected onto x_q. For d dimensions, we can thus replace what was previously $\theta_i^* \in \mathbb{R}$ with $(\theta_i^*)^T x_q \in \mathbb{R}$, where $\theta_i^*, x_q \in \mathbb{R}^d$.

Recall that up to this point we have been assuming that the remaining $\frac{n}{2}$ or $\frac{n}{2} - 1$ points that were added by Alice are *not* support vectors. We will now show that this is the case.

Redefine S_0 to be the set of points including $\frac{n}{4}$ copies of $(x_\alpha, -1)$, $\frac{n}{4}$ copies of $(x_\beta, +1)$, and $\frac{n}{2}$ arbitrary points $\{v_i\}_{i=1}^{n/2}$, but this time with the requirement that $v_i^T x_q < 1 - 10\delta$, $\forall i \in [\frac{n}{2}]$. Similarly, let S_1 be the set of points including $\frac{n}{4}$ copies of $(x_\alpha, -1)$, $\frac{n}{4}$ copies of $(x_\beta, +1)$, one copy of $(x_q, -1)$ and $\frac{n}{2} - 1$ arbitrary points $\{v_i\}_{i=1}^{n/2-1}$, all satisfying $v_i^T x_q < 1 - 10\delta$.

Define

$$F_\lambda^{(0)}(\theta, b; \lambda) := \frac{\lambda}{2}(\|\theta\|^2 + b^2) + \frac{1}{n}\sum_{(x,y)\in S_0} \max\{0, 1 - y(\theta^T x + b)\} \tag{24}$$

and

$$F_\lambda^{(1)}(\theta, b; \lambda) := \frac{\lambda}{2}(\|\theta\|^2 + b^2) + \frac{1}{n}\sum_{(x,y)\in S_1} \max\{0, 1 - y(\theta^T x + b)\} \tag{25}$$

Finally, let $(\theta_i^*, b_i^*) = \arg\min_{\theta,b} F_\lambda^{(i)}(\theta, b)$ for $i \in \{0, 1\}$.

▶ **Lemma 15.** *Suppose $\lambda = \delta^2$, $\delta < \frac{1}{7}$ and $n \geq \frac{1}{\delta^2} = \frac{1}{\lambda}$. Then for both S_0 and S_1, none of the points v satisfying $v^T x_q < 1 - 10\delta$ are support vectors. Moreover, we have*

$$\theta_0^* = \left(\frac{2\lambda(1+\delta)+\delta}{2\lambda(1+(1+\delta)^2)}\right) x_q \tag{26}$$

$$\theta_1^* = \left(\frac{2\lambda(1+\delta)+\delta(1+\frac{1}{n})}{2\lambda(1+(1+\delta)^2)}\right) x_q \tag{27}$$

and $b_i^* = 1 - \|\theta_i^*\|(1+\delta)$ *for both* $i \in \{0,1\}$.

Proof. By the preceding discussion and lemmas, it suffices to show that for $(v,-1) \in S_i$ such that $v^T x_q < 1 - 10\delta$, $(v,-1)$ is not a support vector. This implies that (θ_i^*, b_i^*) does not depend on such v, so that the equations for the optimal parameters from before (adapted slightly for $d > 1$) indeed hold. For this, it suffices to show that for the (θ_i^*, b_i^*) described above, we have $1 + ((\theta_i^*)^T v + b_i^*) < 0$ when $v^T x_q < 1 - 10\delta$.

For the given δ, λ, and n, one can easily verify that for both $i = 0$ and $i = 1$ we have $\|\theta_i^*\| \geq \frac{\delta}{5\lambda}$. Moreover, since $b_i^* = 1 - (1+\delta)\|\theta_i^*\|$ and $(\theta_i^*)^T x_q = \|\theta_i^*\|$, we have:

$$1+(\theta_i^*)^T v+b_i^* \leq 1+(1-10\delta)\|\theta_i^*\|+1-(1+\delta)\|\theta_i^*\| = 2-11\delta\|\theta_i^*\| \leq 2-11\delta\frac{\delta}{5\lambda} = 2-\frac{11}{5} < 0$$

proving the claim. ◀

3.2 Proofs of Main Results

We now use the preceding lemmas to complete the proof of Theorem 12.

Proof. Let $n = \frac{1}{20\sqrt{\varepsilon}}$, and let x_q be the point being queried by Bob. If x_q was added to the set by Alice, then $i := b_q = 0$; otherwise, $i := b_q = 1$. Let $(\hat{\theta}, \hat{b})$ be the output of the sketch. Then using the guarantee of the sketch and of strong convexity, we have that $\|(\hat{\theta}, \hat{b}) - (\theta_i^*, b_i^*)\| \leq \sqrt{\frac{2\varepsilon}{\lambda}}$. Hence, to distinguish these two scenarios, it suffices to show that $\|(\theta_0^*, b_0^*) - (\theta_1^*, b_1^*)\| > 2\sqrt{\frac{2\varepsilon}{\lambda}}$. Indeed,

$$\|(\theta_0^*, b_0^*) - (\theta_1^*, b_1^*)\| \geq \|\theta_0^* - \theta_1^*\| = \frac{1}{n}\frac{\delta}{2\lambda(1+(1+\delta)^2)} \geq \frac{\delta}{5\lambda n} = \frac{20\delta\sqrt{\varepsilon}}{5\lambda} = 4\sqrt{\frac{\varepsilon}{\lambda}} > 2\sqrt{\frac{2\varepsilon}{\lambda}},$$

proving the claim. ◀

Now we extend this result to the low-dimensional case. When d is a constant, the above lower bound cannot be directly applied because there does not exist a set T of size $|T| = n$ such that $\forall v \neq v' \in T$, $v^T v' < 1 - 10\delta$ when δ is a constant. However, we can adapt the lower bound to the low dimensional setting if we let δ be sub-constant. Since we always maintain the relationship that $\lambda = \delta^2$, this means that λ also scales with $\frac{1}{n}$. Note that $\lambda = \Theta(\frac{1}{n})$ is often used in practice.

▶ **Theorem 16.** *For* $d = 2$ *and* $\lambda = \Theta(\frac{1}{n^2})$*, a sketch as defined above requires space* $\Omega(\varepsilon^{-1/4})$*. For* $d \geq 3$ *and* $\lambda = \Theta(\frac{1}{n})$*, such a sketch requires space* $\Omega(\varepsilon^{-1/2})$*.*

Proof. First, note that if $n = O(\delta^{-(d-1)})$ for $d > 1$, then there exists a subset T of the unit sphere in $\mathbb{R}^d$ satisfying $\forall v \neq v' \in T$, $v^T v' \leq 1 - 10\delta$. Alice will use such a set to encode bits.

In the high dimensional lower bound construction, the constraints we had on the points were $\delta < \frac{1}{7}$ and $n \geq \frac{1}{\delta^2} = \frac{1}{\lambda}$. Moreover, for the final step of the analysis, we needed that $\frac{\delta}{5\lambda n} = \frac{1}{5\delta n} > 2\sqrt{\frac{2\varepsilon}{\lambda}} = 2\sqrt{2\varepsilon}\frac{1}{\delta}$, which is always satisfied for $n \leq \frac{1}{20\sqrt{\varepsilon}}$ (regardless of λ and δ, as long as $\lambda = \delta^2$). Additionally, for the construction we must also have $n \leq \Theta(\delta^{-(d-1)})$. Hence, for $d = 3$, by letting $\lambda = \frac{1}{n} = \Theta(\varepsilon^{1/2})$, all of these constraints are satisfied, so that we again get a $\frac{1}{\sqrt{\varepsilon}}$ lower bound. For $d = 2$ and $\lambda = \Theta(\varepsilon^{1/2})$, this also implies that we can encode $\Theta(\delta^{-(d-1)}) = \Theta(\varepsilon^{-1/4})$ bits. ◀

References

1 Alekh Agarwal, Peter Bartlett, Pradeep Ravikumar, and Martin J. Wainwright. Information-theoretic lower bounds on the oracle complexity of convex optimization. *International Conference on Neural Information Processing Systems (NIPS)*, 2009.

2 Zeyuan Allen-Zhu. Katyusha: The first direct acceleration of stochastic gradient methods. *Symposium on Theory of Computing (STOC)*, 2017.

3 Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. On the fine-grained complexity of empirical risk minimization: Kernel methods and neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, 2017.

4 Kenneth L. Clarkson and David P. Woodruff. Numerical linear algebra in the streaming model. *Symposium on Theory of Computing (STOC)*, 2009.

5 J. H. Huggins, R. P. Adams, and T. Broderick. PASS-GLM: polynomial approximate sufficient statistics for scalable Bayesian GLM inference. *International Conference on Neural Information Processing Systems (NIPS)*, 2017.

6 T. S. Jayram and David P. Woodruff. Optimal bounds for johnson-lindenstrauss transforms and streaming problems with subconstant error. *ACM Transactions on Algorithms*, 2013.

7 Yi Li, Ruosong Wang, and David P. Woodruff. Tight bounds for the subspace sketch problem with applications. In *Proceedings of SODA*, 2020.

8 Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *International Conference on Neural Information Processing Systems (NIPS)*, 2007.

9 Piyush Rai, Hal Daumé III, and Suresh Venkatasubramanian. Streamed learning: One-pass svms. *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

10 Mark Schmidt, Nicolas Le Roux, and Francis Bach. Minimizing finite sums with the stochastic average gradient. *Mathematical Programming*, 2017.

11 Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

12 Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. Pegasos: Primal estimated sub-gradient solver for svm. *International Conference on Machine Learning (ICML)*, 2007.

13 Ivor W. Tsang, James T. Kwok, and Pak-Ming Cheung. Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research (JMLR)*, 2005.

A Analysis of the streaming algorithm for $(1+\varepsilon)$-multiplicative approximation in the case of $d = 1$

First, it is clear from the algorithm description that in the case of $q \le p$, the algorithm produces an exact solution. To show the correctness in the otherwise case, as stated in the description let t_j be the number of points in P that fall in the interval R_j, and let T_j be the number of points that fall to the left of R_j, i.e., $T_j = |\{x \in P\colon x \le q - (D/2^{j-1})\}| = |\bigcup_{k<j} R_k|$ and note that $T_j \ge \max\{|S_0|, |E_0|\} \ge \frac{2\log W}{\varepsilon}\max\{\log W + 1/\varepsilon\}$. Moreover, let $V = \sum_{x\in P, x\le q}(q-x)$ be the value of the solution and $V_j = \sum_{x\in P\cap R_j}(q-x)$ be the contribution of the points in R_j to the solution.

▶ **Lemma 17** (If Not Enough Samples, Set Doesn't Contribute). *If $t_j < \log D$, then $i'(j) = -1$. Moreover, if $i'(j) = -1$, then with high probability $\frac{t_j}{T_j} \le \frac{2\varepsilon}{\log W}$.*

Proof. For the first claim, note that for such j, even if we sample the points with probability 1, we don't get $\log D$ points in E_i, and thus $i'(j) = -1$.

Now suppose $i'(j) = -1$. If $t_j \le 2\log D$, then since $T_j \ge \log^2 W/\varepsilon$, we have that $\frac{t_j}{T_j} \le \frac{2\varepsilon}{\log W}$. Otherwise, let i' be such that $2^{-(i'+1)} \le \frac{2\log D}{t_j} \le 2^{-i'}$. Now, if we sample every point in R_j with probability $2^{-i'}$, in expectation, we sample $t_j 2^{-i'}$ points which is between $2\log D$ and $4\log D$. Moreover, since $t_j \ge 2\log D$, we can use Chernoff bound, proving that

with high probability the number of samples is between $\log D$ and $8 \log D$. Therefore, since the final sample $E_{i'}$ does not contain $\log D$ points from R_j, it means that it contains at least $m_1 - 8 \log D \geq m_1/2 = \frac{C_1 \log^2 W}{2\varepsilon}$ points from $\cup_{k<j} R_k$ (for sufficiently large constant C_1).

On the other hand, the expected number of sampled points from $\bigcup_{k<j} R_k$ is $T_j 2^{-i'}$ which with high probability (using Chernoff again) should be at least $m_1/4 \geq \frac{C_1 \log^2 W}{4\varepsilon}$. Therefore, we get that

$$T_j \geq 2^{i'} \frac{C_1 \log^2 W}{4\varepsilon} \geq \frac{t_j}{4 \log D} \cdot \frac{C_1 \log^2 W}{4\varepsilon} \geq \frac{C_1 t_j \log W}{16\varepsilon}$$

Thus for $C_1 \geq 8$, we get that $\frac{t_j}{T_j} \leq \frac{2\varepsilon}{\log W}$ ◀

▶ **Lemma 18** (If Enough Samples, Get Initial Constant Factor Approximation). *If $i'(j) \neq -1$, then ϕ_j approximates $\min\{1, \frac{t_j}{T_j}\}$ by a constant factor.*

Proof. By Lemma 17, we know that $t_j \geq \log D$.

- First note that if $T_j \leq t_j$, with high probability the number of sampled points from $\cup_{k<j} R_k$ is less than 2 times the number of sampled points from R_j, and therefore, $\phi_j \geq 1/2$, and the lemma is proved.
- Second, note that if $\frac{t_j}{T_j} \leq \frac{\varepsilon}{2C_1 \log W} \cdot \frac{\log D}{\log W}$, then the number of sampled points from $\cup_{k<j} R_k$ with high probability (using Chernoff) is at least $\frac{1}{2} \cdot \frac{T_j}{t_j} \cdot \log D \geq \frac{\log D}{2} \cdot \frac{2C_1 \log^2 W}{\log D \varepsilon} \geq \frac{C_1 \log^2 W}{\varepsilon} = m_1$ which is a contradiction, because then we would not have picked $\log D$ points in $E_{i'(j)}$ from R_j.
- In the otherwise case, we show that the sample $E_{i'(j)}$ suffices to get a constant factor approximation to both values t_j and T_j (and hence, t_j/T_j). First, to see the latter, note that $T_j \geq t_j$ and therefore with high probability (using Chernoff) we get at least $\frac{\log D}{2}$ samples from $\cup_{k<j} R_k$ in the set $E_{i'(j)}$ which are chosen uniformly at random. This is enough for computing a constant factor approximation of T_j with high probability.
  Second, if $E_{i'(j)}$ only contains samples from the first $t_j/8$ fraction of the points in R_j, then with high probability, there would still be $\log D$ samples from R_j in $E_{i'(j)+1}$ (this is because with high probability, we only get less points from $\bigcup_{k<j} R_k$ in $E_{i'(j)+1}$ and still at least $\log D$ points from R_j in it). However, this contradicts the choice of $i'(j)$. Therefore, we get a uniform sample of size $\Omega(\log D)$, from a constant fraction of the points in R_j, meaning that we can approximate t_j up to a constant factor with high probability. Therefore, ϕ_j will be a constant approximation to the value of $\min\{1, \frac{t_j}{T_j}\}$. ◀

The following two lemmas analyze the case of $\phi_j \geq \frac{1}{\log W}$.

▶ **Lemma 19** (Enough Samples are Found From Large Contributing Sets). *If $i'(j) \neq -1$ and $\phi_j \geq \frac{1}{\log W}$, then we have $i(j) \neq -1$.*

Proof. Note that by Lemma 18, ϕ_j is a constant factor approximation for $\min\{1, t_j/T_j\}$ which means that either $t_j \geq \Omega(T_j)$ or $t_j \geq \Omega(T_j/\log W)$ as $\phi_j \geq 1/\log W$. Let C_3 be the constant in this inequality, i.e., $t_j \geq T_j/(C_3 \log W)$. Moreover, as $T_j \geq |S_0| = \frac{C_2 \log W}{\varepsilon^2}$, we get that $t_j \geq \frac{C_2}{C_3 \varepsilon^2} \geq 2/\varepsilon^2$ for $C_2 \geq 2C_3$.

Now let i be such that $2^{-(i+1)} \leq 2/(\varepsilon^2 t_j) \leq 2^{-i}$. This means that by sampling the points with rate 2^{-i}, in expectation, we sample $2^{-i} t_j \geq 2/\varepsilon^2$ points from R_j and moreover, with high probability we will sample at least $1/\varepsilon^2$ points from R_j (here we used the fact that since $i'(j) \neq -1$, we have $t_j \geq \log D$ and thus we can apply Chernoff bound). Furthermore,

in expectation, we will sample $2^{-i}T_j \leq 4T_j/(t_j\varepsilon^2) \leq 4C_3 \cdot \log W/\varepsilon^2$ points from intervals $R_0, \ldots, R_{j-1}$. Therefore, since we keep $m_2 = C_2 \log W/\varepsilon^2$ smallest sampled points in S_i, by choosing C_2 large enough, i.e., $C_2 \geq (4C_3 + 2)$, we will store all sampled points of R_j in S_i as well. Therefore, $i(j) \neq -1$. ◀

▶ **Lemma 20** (Large Contributing Sets have Small Relative Error). *If $i'(j) \neq -1$ and $\phi_j \geq \frac{1}{\log W}$, then we have that $\sum_{x \in S_{i(j)} \cap R_j} 2^i(q-x)$ is a $(1+2\varepsilon)$ approximation of V_j.*

Proof. As in the previous lemma, let i be such that $2^{-(i+1)} \leq 2/(\varepsilon^2 t_j) \leq 2^{-i}$. Therefore, by similar arguments to the above lemma, we know that $i(j) \geq i$, which further means that all sampled points from R_j are kept in $S_{i(j)}$. We thus get a uniform sample of size at least $1/\varepsilon^2$ from the interval R_j, which is enough for an additive $\varepsilon t_j(D/2^{j-1})$ approximation as every point in the interval is contributing at least $D/2^{j-1}$. On the other hand, we have that $V_j \geq t_j D/2^j$. Thus this additive approximation translates to a $(1+2\varepsilon)$ multiplicative approximation. ◀

The following two lemmas analyze the case of $\frac{\varepsilon}{\log W} \leq \phi_j \leq \frac{1}{\log W}$.

▶ **Lemma 21** (Enough Samples are Found from Small Contributing Sets). *If $i'(j) \neq -1$ and $\frac{\varepsilon}{\log W} \leq \phi_j \leq \frac{1}{\log W}$, then if $V_j \geq V(\varepsilon/\log W)$, we have $i(j) \neq -1$.*

Proof. Note that $V_j \geq V(\varepsilon/\log W)$ means that $t_j \geq T_j(\varepsilon/\log W)$, as otherwise if $t_j < T_j(\varepsilon/\log W)$, we have that the total contribution of R_j is at most $V_j \leq t_j D/2^{j-1}$, while we have that $V \geq T_j D/2^{j-1}$; which means that $V_j \leq T_j(\varepsilon/\log W)D/2^{j-1} \leq (\varepsilon/\log W)V$, which is a contradiction.

Now let i be such that $2^{-(i+1)} \leq 2(\phi_j \log W)^2/(\varepsilon^2 t_j) \leq 2^{-i}$ [3]. This means that in expectation, we sample $2^{-i}t_j \geq 2(\phi_j \log W/\varepsilon)^2$ points from R_j and moreover, with high probability we will sample at least $(\phi_j \log W/\varepsilon)^2$ points from R_j. Furthermore, in expectation, we will sample $2^{-i}T_j \leq 4T_j(\phi_j \log W)^2/(t_j\varepsilon^2) \leq 4\phi \log^2 W/\varepsilon^2 \leq 2\log W/\varepsilon^2$ points from intervals $R_0, \ldots, R_{j-1}$, where in the last inequality we used the fact that $\phi_j \leq 1/\log W$. Therefore, since we keep $m_2 \geq 6\log W/\varepsilon^2$ smallest points in S_i, we will store all sampled points of R_j in S_i. Therefore, $i(j) \neq -1$. ◀

▶ **Lemma 22** (Small Contributing Sets have Small Additive Error). *If $i'(j) \neq -1$ and $\frac{\varepsilon}{\log W} \leq \phi_j \leq \frac{1}{\log W}$, then if $V_j \geq V(\varepsilon/\log W)$, we have that $\sum_{x \in S_{i(j)} \cap R_j} 2^i(q-x)$ is a $\frac{\varepsilon}{\log W}$ additive approximation of V.*

Proof. As in the previous lemma, let i be such that $2^{-(i+1)} \leq 2(\phi_j \log W)^2/(\varepsilon^2 t_j) \leq 2^{-i}$. Therefore, by similar arguments to the above lemma, we know that $i(j) \geq i$, which further means that all sampled points from R_j are kept in $S_{i(j)}$. We thus get a uniform sample of size at least $(\phi_j \log W/\varepsilon)^2$ from the interval R_j, which is enough for an additive $\frac{\varepsilon}{\phi_j \log W} t_j(D/2^{j-1})$ approximation as every point in the interval is contributing at least $D/2^{j-1}$. However, as $V \geq T_j D/2^{j-1}$, this translates to a $\frac{\varepsilon}{\phi_j \log W}\frac{t_j}{T_j}V = \frac{\varepsilon}{\log W}V$ additive approximation. ◀

We now restate and prove Lemma 5 and Corollary 6.

▶ **Lemma 23** (Lemma 5). *The algorithm returns a $(1+O(\varepsilon))$ multiplicative approximation.*

[3] Note that again by the conditions of the lemma on ϕ_j, the fact that ϕ_j approximates t_j/T_j, and the fact that $T_j \geq C_2 \log W/\varepsilon^2$, this value is always at most 1 and therefore such i exists.

Proof. Let us consider the following cases separately.

- For $j = 0$, we get the exact contribution of the points in R_0.
- For j where $i'(j) = -1$, we know by lemma 17 that $\frac{t_j}{T_j} \leq 2\varepsilon/\log W$. Therefore the total contribution of all V_j for such j is at most $2\varepsilon V$.
- Moreover, whenever $i(j) = -1$, using Lemmas 19 and 21, we know that $V_j \leq V(\varepsilon/\log W)$. Summing over all such j, we get a total additive error of εV.
- Now consider all $j \geq 1$ such that $\phi_j \geq \frac{1}{\log W}$. By Lemma 19 and Lemma 20, we get a $(1 + 2\varepsilon)$ multiplicative approximation of their contribution.
- Finally consider all $j \geq 1$ such that $\frac{\varepsilon}{\log W} \leq \phi_j \leq \frac{1}{\log W}$. If $V_j \geq V(\varepsilon/\log W)$, by lemma 22, we get an additive $\frac{\varepsilon}{\log W} V$ approximation of their contribution. Summing over all such j, this will give a $(1 + \varepsilon)$ multiplicative approximation.
  However if $V_j < V(\varepsilon/\log W)$ and $i(j) \neq -1$ for such j, it means that we have sampled points in R_j with probability $2^{-i(j)}$ and potentially kept only some of them in $S_{i(j)}$ this only causes an under estimation of the contribution of R_j. However, the samples that the algorithm has chosen to keep in S_i might be biased towards the smaller end of the interval, i.e., $D/2^{j-1}$, however this can only cause an over estimation by a factor of 2. Therefore, for such a j, we have $0 \leq \sum_{x \in S_{i(j)} \cap R_j} 2^i(q - x) \leq 2(1 + \varepsilon)V_j$. Again because $V_j \leq V\varepsilon/\log W$, the total error of such j will be at most a multiplicative $(1 + \varepsilon)$ factor. ◀

▶ **Corollary 24** (Corollary 6). *There exists a one pass streaming algorithm that computes a $(1 + \varepsilon)$ multiplicative approximation for point estimation variant of the problem in one dimensional case. Moreover, if the points come from $[W]$, the space usage of the algorithm is $O(\frac{\log^2 n \cdot \log W}{\varepsilon}(\log n + 1/\varepsilon))$ bits.*

Proof. Note that in the above algorithm we do not need to consider R_j for which $j \geq \log n^2 = 2 \log n$, as the overall contribution of such points to the solution is as most D/n whereas the value of the solution is at least D. Thus we can bound one of the $\log W$ in the bound of Observation 4 by $\log n$. ◀

B Multiplicative point estimation lower bound

Proof. We prove this theorem by a reduction from the standard Augmented Indexing problem. In this problem, Alice is given a bit string s of length m and Bob has an index $i \in [m]$ as well as bits $s_1, \ldots, s_{i-1}$. The goal is for Alice to send a message to Bob so that he can recover s_i, the ith bit in Alice's string. It is a standard fact that this requires $\Omega(m)$ bits.

Case of $d = 1$. First consider the one dimensional case and let $r = 1/\sqrt{\varepsilon}$. Suppose that Alice holds a string of length r, termed $s_1 \ldots s_r$. From that, she will construct an instance of our point estimation sketching problem. For each $0 \leq i < r$, if her ith bit is one, she will put $n/r = n\sqrt{\varepsilon}$ points in position $3i/r = 3i\sqrt{\varepsilon}$. Otherwise if the bit is 0, she will not put any point there. Thus all points will be positioned in $[0, 1)$ with the diameter less than 1.

To learn the ith bit, Bob will query the presumed sketch with $b = 3(i + 1)\sqrt{\varepsilon}$, obtaining a value v. Bob will subtract the contribution from the points associated with the first $i - 1$ bits. Note that the resulting value is $\frac{1}{n} \cdot s_i \cdot n\sqrt{\varepsilon} \cdot 3\sqrt{\varepsilon}$, up to an additive error ε. Therefore it is possible to recover the value of the encoded bit. Hence the one-dimensional point estimation problem cannot be solved in space less than $\Omega(r) = \Omega(1/\sqrt{\varepsilon})$.

Case of $d = 2$. Next, consider the two dimensional case, and for parameters s and r (to be specified later), consider the $s \times r$ potential positions inside a circle of unit radius. More specifically, for $1 \le i \le s, 1 \le j \le r$, the (i, j)-th position is the point at angle $2\pi i/s$ and at radius $1 - \frac{j-1}{2r}$. These positions correspond to the sr bits in the index problem held by Alice. For the $((j-1)s+i)$-th bit in her bit-string, Alice will put $n/(sr)$ actual points at the (i, j)-th position described above iff the corresponding bit is equal to 1. She will then send her point set to Bob.

Bob can recover any bit of Alice using hyperplane queries. Specifically, in order to figure out the $((j-1)s+i)$-th bit of Alice, Bob can ask the hyper-plane corresponding to $\theta = 2\pi i/s$ and $b = 1 - j/(2r)$; and subtract the contribution of points corresponding to bits up to $((j-1)s+i)$. Note that, if there is no point at that location, the result should be 0, otherwise it should be $(n/(sr)) \cdot (1/(2r))$. Thus, if the algorithm has an additive approximation less than $\frac{1}{3}(n/(sr)) \cdot (1/(2r))$, it can correctly recover the respective bit in Alice's input.

Also, note that the above means that no multiplicative approximation is possible unless Alice sends s bits. But setting $s = n$ and $r = 1$, this will require Alice to send all her input to Bob.

Note that for the hyperplane to include only the ith point from the tier j circle, we need to set r so that $1/(2r) \approx (1 - \cos(2\pi/s)) \approx 2\pi^2/s^2$. Thus we need additive approximation $\Theta(n/s^5)$. We set $s = \varepsilon^{-1/5}$, and get that the total size of the sketch is at least $\Omega(sr) = \Omega(\varepsilon^{-3/5})$.

Case of $d > 2$. To generalize the result to higher dimensions, we put s points uniformly on the $d-1$ dimensional unit sphere and repeat this for r different radii as before. More precisely, using we put an ℓ-net on the surface of the unit sphere. It is a standard fact that we can have $s = \Theta(1/\ell)^{d-1}$ points on the surface so that their pairwise distance is at least ℓ. Similar to the two-dimensional case, we have that $(1/2r) \approx (1 - \cos(O(\pi/\ell))) \approx O(1/\ell^2)$. Therefore, the additive approximation the algorithm can tolerate is $\frac{n}{sr} \cdot \frac{1}{2r}$ which should be at most $n\varepsilon$ and therefore, we get that $\varepsilon = 1/(sr^2)$. Inserting the values of s and r using the value of ℓ, we get that $\varepsilon = \ell^{d+3}$. As the space lower bound for the index problem is $\Omega(sr)$, we get that the space requirement for our problem is $sr = 1/(\varepsilon r) = \ell^2/\varepsilon = \varepsilon^{2/(d+3)}/\varepsilon = \varepsilon^{-\frac{d+1}{d+3}}$. ◀

C Additive point estimation upper bounds

C.1 $d = 1$

Proof. We now argue the correctness and space complexity of the algorithm given in Theorem 9. For correctness, the output of a query q accounts for all the data points entirely to the left of the interval $I_{v(q)}$, where $v(q)$ is the unique leaf $v(q)$ with $q \in I_{v(q)}$. Hence the error comes entirely from the unaccounted points in $v(q)$ as well as the predecessors of $v(q)$. By construction any (non-empty) predecessor has diameter $\sqrt{\varepsilon}$ and contains less than $\sqrt{\varepsilon}n$ points, or, alternatively, has diameter less than ε (for expansion exception). Hence the error is $\le \varepsilon$ for each predecessor, and $O(\varepsilon \log 1/\varepsilon)$ overall. As usual, we can rescale ε to obtain error ε and correspondingly larger space.

For space complexity, note that the space is proportional to the size of the tree. The tree has size at most $O(1/\sqrt{\varepsilon})$ since each leaf is either one of the original $2/\sqrt{\varepsilon}$ one or has a parent node with $\sqrt{\varepsilon}n$ points. ◀

C.2 $d = 2$

Proof. We now analyze the procedure described in Theorem 10. It is clear that the total space usage of the algorithm is at most order of the size of the quad-tree. The size of the tree is bounded by $O(1/\varepsilon)$ as follows. First, there at most $O(1/\varepsilon)$ of the original nodes. Second, each new children created has the property that its parent got associated with εn points, hence at most $4/\varepsilon$ such children can be ever created.

We now analyze the error of the sketching algorithm. For the nodes that do not cross the line L, the distances of their points are computed exactly. So we only need to argue about the crossing nodes. First of all, note that we can ignore all leaves with more than εn points at them as their diameter is less than $2\varepsilon^2$.

Let $\mathcal{C}$ be the set of leaves that cross the query line and have diameter at least $2\varepsilon^2$ (and hence less than εn associated points). It is immediate to check that, in expectation, our estimator outputs the correct value; in particular for P_v the set of points associated to a node v:

$$\mathbb{E}\left[\sum_{v\in\mathcal{C}} c_v \cdot \max\{0, D(r_v, L)\}\right] = \sum_{v\in\mathcal{C}}\sum_{p\in P_v} \frac{1}{c_v} c_v \cdot \max\{0, D(p, L)\} = \sum_{v\in\mathcal{C}}\sum_{p\in P_v} \max\{0, D(p, L)\}.$$

Thus we only need to argue that it concentrates closely around its expectations, with constant probability. Let us compute the variance. The point in each (non-empty) node is chosen independently at random. Thus we can sum up the variances of each node. Consider a node v with side length $\ell \geq \varepsilon^2$. Then we have that

$$\mathrm{Var}\left[c_v \cdot \max\{0, D(r_v, L)\}\right] \leq \sum_{i=1}^{c_v} \frac{1}{c_v} \cdot (c_v \cdot \ell)^2 \leq c_v^2 \ell^2 \leq (n\varepsilon)^2 \ell^2.$$

Now note that any line can intersect only $(1/\ell)$ nodes with side length ℓ, and thus the total variance of all nodes with side length ℓ is at most $(n\varepsilon)^2\ell$. Hence, the total variance over all nodes (over all levels) is at most $O(n^2\varepsilon^2\sqrt{\varepsilon})$. Overall, the standard deviation is at most $O(n\varepsilon^{5/4})$. By Chebyshev's bound, the reported answer has an additive error of $O(n\varepsilon^{5/4})$ with constant probability.

As stated before, replacing $\varepsilon' = \Theta(\varepsilon^{4/5})$ we get that the algorithm is providing an additive ε' approximation using space $\tilde{O}((\varepsilon')^{-4/5})$, completing the proof of the result. ◀

On the Parameterized Approximability of Contraction to Classes of Chordal Graphs

Spoorthy Gunda
Simon Fraser University, Burnaby, Canada
sgunda@sfu.ca

Pallavi Jain
Indian Institute of Technology Jodhpur, India
pallavi@iitj.ac.in

Daniel Lokshtanov
University of California, Santa Barbara, CA, USA
daniello@ucsb.edu

Saket Saurabh
The Institute of Mathematical Sciences, HBNI, Chennai, India
University of Bergen, Norway
saket@imsc.res.in

Prafullkumar Tale
Max Planck Institute for Informatics, Saarland Informatics Campus, Saarbrücken, Germany
prafullkumar.tale@mpi-inf.mpg.de

Abstract

A graph operation that *contracts edges* is one of the fundamental operations in the theory of graph minors. Parameterized Complexity of editing to a family of graphs by contracting k edges has recently gained substantial scientific attention, and several new results have been obtained. Some important families of graphs, namely the subfamilies of chordal graphs, in the context of edge contractions, have proven to be significantly difficult than one might expect. In this paper, we study the $\mathcal{F}$-Contraction problem, where $\mathcal{F}$ is a subfamily of chordal graphs, in the realm of parameterized approximation. Formally, given a graph G and an integer k, $\mathcal{F}$-Contraction asks whether there exists $X \subseteq E(G)$ such that $G/X \in \mathcal{F}$ and $|X| \leq k$. Here, G/X is the graph obtained from G by contracting edges in X. We obtain the following results for the $\mathcal{F}$-Contraction problem.

- Clique Contraction is known to be FPT. However, unless NP $\subseteq$ coNP/poly, it does not admit a polynomial kernel. We show that it admits a polynomial-size approximate kernelization scheme (PSAKS). That is, it admits a $(1+\epsilon)$-approximate kernel with $\mathcal{O}(k^{f(\epsilon)})$ vertices for every $\epsilon > 0$.
- Split Contraction is known to be W[1]-Hard. We deconstruct this intractability result in two ways. Firstly, we give a $(2+\epsilon)$-approximate polynomial kernel for Split Contraction (which also implies a factor $(2+\epsilon)$-FPT-approximation algorithm for Split Contraction). Furthermore, we show that, assuming Gap-ETH, there is no $\left(\frac{5}{4}-\delta\right)$-FPT-approximation algorithm for Split Contraction. Here, $\epsilon, \delta > 0$ are fixed constants.
- Chordal Contraction is known to be W[2]-Hard. We complement this result by observing that the existing W[2]-hardness reduction can be adapted to show that, assuming FPT$\neq$ W[1], there is no $F(k)$-FPT-approximation algorithm for Chordal Contraction. Here, $F(k)$ is an arbitrary function depending on k alone.

We say that an algorithm is an $h(k)$-FPT-approximation algorithm for the $\mathcal{F}$-Contraction problem, if it runs in FPT time, and on any input (G,k) such that there exists $X \subseteq E(G)$ satisfying $G/X \in \mathcal{F}$ and $|X| \leq k$, it outputs an edge set Y of size at most $h(k) \cdot k$ for which G/Y is in $\mathcal{F}$. We find it extremely interesting that three closely related problems have different behavior with respect to FPT-approximation.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis

Keywords and phrases Graph Contraction, FPT-Approximation, Inapproximability, Lossy Kernels



© Spoorthy Gunda, Pallavi Jain, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale;
licensed under Creative Commons License CC-BY

Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM 2020).
Editors: Jarosław Byrka and Raghu Meka; Article No. 51; pp. 51:1–51:19



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

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.51

Category APPROX

Related Version A full version of the paper can be found here `https://arxiv.org/abs/2006.10364`, [31].



Funding *Saket Saurabh*: 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 819416), and Swarnajayanti Fellowship (No DST/SJF/MSA01/2017-18).
Prafullkumar Tale: This research is a part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme under grant agreement SYSTEMATICGRAPH (No. 725978). Most part of this project was completed when the author was a Senior Research Fellow at The Institute of Mathematical Sciences, HBNI, Chennai, India.

1 Introduction

Graph modification problems have been extensively studied since the inception of Parameterized Complexity in the early '90s. The input of a typical graph modification problem consists of a graph G and a positive integer k, and the objective is to edit k vertices (or edges) so that the resulting graph belongs to some particular family, $\mathcal{F}$, of graphs. These problems are not only mathematically and structurally challenging, but have also led to the discovery of several important techniques in the field of Parameterized Complexity. It would be completely appropriate to say that solutions to these problems played a central role in the growth of the field. In fact, just in the last few years, parameterized algorithms have been developed for several graph editing problems [12, 10, 11, 9, 5, 6, 26, 24, 17, 18, 19, 27]. The focus of all of these papers and the vast majority of papers on parameterized graph editing problems has so far been limited to edit operations that delete vertices, delete edges or add edges.

In recent years, a different edit operation has begun to attract significant scientific attention. This operation, which is arguably the most natural edit operation apart from deletions/insertions of vertices/edges, is the one that contracts an edge. Here, given an edge uv that exists in the input graph, we remove the edge from the graph and merge its two endpoints. Edge contraction is a fundamental operation in the theory of graph minors. For some particular family of graphs, $\mathcal{F}$, we say that a graph G belongs to $\mathcal{F} + kv$, $\mathcal{F} + ke$ or $\mathcal{F} - ke$ if some graph in $\mathcal{F}$ can be obtained by deleting at most k vertices from G, deleting at most k edges from G or adding at most k edges to G, respectively. Using this terminology, we say that a graph G belongs to $\mathcal{F}|ke$ if some graph in $\mathcal{F}$ can be obtained by contracting at most k edges in G. In this paper, we study the following problem.

$\mathcal{F}$-Contraction **Parameter:** k
Input: A graph G and an integer k
Question: Does G belong to $\mathcal{F}|ke$?

For several families of graphs $\mathcal{F}$, early papers by Watanabe et al. [46, 47], and Asano and Hirata [3] showed that $\mathcal{F}$-Edge Contraction is NP-complete.

In the framework of Parameterized Complexity, these problems exhibit properties that are quite different from those problems where we only delete or add vertices and edges. Indeed, a well-known result by Cai [7] states that in case $\mathcal{F}$ is a hereditary family of graphs

with a finite set of forbidden induced subgraphs, then the graph modification problems, $\mathcal{F}+kv$, $\mathcal{F}+ke$ or $\mathcal{F}-ke$, defined by $\mathcal{F}$ admits a simple FPT algorithm (an algorithm with running time $f(k)n^{\mathcal{O}(1)}$). However, for $\mathcal{F}$-Contraction, the result by Cai [7] does not hold. In particular, Lokshtanov et al. [38] and Cai and Guo [8] independently showed that if $\mathcal{F}$ is either the family of P_ℓ-free graphs for some $\ell \geq 5$ or the family of C_ℓ-free graphs for some $\ell \geq 4$, then $\mathcal{F}$-Contraction is W[2]-Hard (W[i]-hardness, for $i \geq 1$, is an analogue to NP-hardness in Parameterized Complexity, and is used to rule out FPT-algorithm for the problem) when parameterized by k (the number of edges to be contracted). These results immediately imply that Chordal Contraction is W[2]-Hard when parameterized by k. The parameterized hardness result for Chordal Contraction led to finding subfamilies of chordal graphs, where the problem could be shown to be FPT. Two subfamilies that have been considered in the literature are families of *split graphs* and *cliques*. Cai and Guo [8] showed that Clique Contraction is FPT, however, it does not admit a polynomial kernel. Later, Cai and Guo [32] also claimed to design an algorithm that solves Split Contraction in time $2^{\mathcal{O}(k^2)} \cdot n^{\mathcal{O}(1)}$, which proves that the problem is FPT. However, Agrawal et al. [2] found an error with the proof and showed that Split Contraction is W[1]-Hard.

Inspired by the intractable results that Chordal Contraction, Split Contraction and Clique Contraction are W[2]-Hard, W[1]-Hard, and does not admit polynomial kernel, respectively, we study them from the viewpoint of parameterized approximation.

Our Results and Methods. We start by defining a few basic definitions in parameterized approximation. To formally define these, we need a notion of parameterized optimization problems. We defer formal definitions to Section 2 and give intuitive definitions here. We say that an algorithm is an $h(k)$-FPT-approximation algorithm for the $\mathcal{F}$-Contraction problem, if it runs in FPT time, and on any input (G,k) if there exists $X \subseteq E(G)$ such that $G/X \in \mathcal{F}$ and $|X| \leq k$, it outputs an edge set Y of size at most $h(k) \cdot k$ and $G/Y \in \mathcal{F}$. Let $\alpha \geq 1$ be a real number. We now give an informal definition of α-approximate kernels. The kernelization algorithm takes an instance I with parameter k, runs in polynomial time, and produces a new instance I' with parameter k'. Both k' and the size of I' should be bounded in terms of just the parameter k. That is, there exists a function $g(k)$ such that $|I'| \leq g(k)$ and $k' \leq g(k)$. This function $g(k)$ is called the *size* of the kernel. For minimization problems, we also require the following from α-approximate kernels: For every $c \geq 1$, a c-approximate solution S' to I' can be transformed in polynomial time into a $(c \cdot \alpha)$-approximate solution S to I. However, if the quality of S' is "worse than" k', or $(c \cdot \alpha) \cdot OPT(I) > k$, the algorithm that transforms S' into S is allowed to fail. Here, $OPT(I)$ is the value of the optimum solution of the instance I.

Our first result is about Clique Contraction. It is known to be FPT. However, unless NP $\subseteq$ coNP/poly, it does not admit a polynomial kernel [8]. We show that it admits a PSAKS. That is, it admits a $(1+\epsilon)$-approximate polynomial kernel with $\mathcal{O}(k^{f(\epsilon)})$ vertices for every $\epsilon > 0$. In particular, we obtain the following result.

▶ **Theorem 1.** *For any $\epsilon > 0$, Clique Contraction parameterized by the size of solution k, admits a time efficient $(1+\epsilon)$-approximate polynomial kernel with $\mathcal{O}(k^{d+1})$ vertices, where $d = \lceil \frac{1}{\epsilon} \rceil$.*

Overview of the proof of Theorem 1. Let us fix an input (G, k) and a constant $\epsilon > 0$. Given a graph G, contracting edges of G to get into a graph class $\mathcal{F}$ is same as partitioning the vertex set $V(G)$ into connected sets, $W_1, W_2, \ldots, W_\ell$, and then contracting each connected set to a vertex. These connected sets are called *witness sets*. A witness set W_i is called *non-trivial*, if $|W_i| \geq 2$, and *trivial* otherwise.

Observe that if a graph G can be transformed into a clique by contracting edges in F, then G can also be converted into a clique by deleting all the endpoints of edges in F. This observation implies that if G is *k-contractible* to a clique, then there exists an induced clique of size at least $|V(G)| - 2k$. Let I be a set of vertices in G, which induces this large clique and let $C = V(G) \setminus I$. Observe that C forms a vertex cover in the graph $\overline{G}$ (graph with vertex set $V(G)$ and those edges that are not present in $E(G)$). Using a factor 2-approximation algorithm, we find a vertex cover X of $\overline{G}$. Let $Y = V(\overline{G}) - X$ be an independent set in $\overline{G}$. If $|X| > 4k$, we immediately say No. Now, suppose that we have some solution and let $W_1, W_2, \ldots, W_\ell$ be those witness sets that are either non-trivial or contained in X. Now, let us say that a set W_i is *nice* if it has at least one vertex outside X, and *small* if it contains less than $\mathcal{O}(1/\epsilon)$ vertices. A set that is not *small* is *large*. Observe that there exists a $(1+\epsilon)$-approximate solution where *the only sets* that are not nice are small. Also, observe that all nice sets are adjacent. Now, we classify all subsets of X of size at most $\mathcal{O}(1/\epsilon)$ as *possible* and *impossible* small witness sets. Notice that if a set $A \subseteq X$ has more than $2k$ non-neighbors, then it can not possibly be a witness set, as one of these non-neighbors will be a trivial witness set. Now for every set, $A \subseteq X$ of size at most $\mathcal{O}(1/\epsilon)$ mark all of its non-neighbors, but if there are more than $2k$, then mark $2k+1$ of them. Now, look at an unmarked vertex in Y, the only reason it could still be relevant if it is part of some W_i. So its job is (a) connecting the vertices in W_i, or (b) potentially being the vertex in Y that is making some W_i nice, or (c) it is a neighbor to *all* the small (not nice) subsets of X in the solution. Now notice that any vertex in Y that is unmarked does jobs (b) and (c) equally well. So we only need to care about connectivity. Look at some nice and small set W_i; we only need to preserve the neighborhoods of the vertices of Y into W_i. For every subset of size $\mathcal{O}(1/\epsilon)$, we keep one vertex in Y that has that set in its neighborhood. Notice that we do not care that different W_i's use different marked vertices for connectivity because merging two W_i's is more profitable for us. Finally, we delete all unmarked vertices and obtain an $(1+\epsilon)$-approximate kernel of size roughly $k^{\mathcal{O}(1/\epsilon)}$. We argue that this kernelization algorithm is time efficient i.e. the running time is polynomial in the size of an input and the constant in the exponent is independent of ϵ. This completes the overview of the proof for Theorem 1. Next, we move to Split Contraction.

Split Contraction is known to be W[1]-Hard [2]. We ask ourselves whether Split Contraction is *completely* FPT-*inapproximable* or admits an α-FPT-approximation algorithm, for some fixed constant $\alpha > 0$. We obtain two results towards our goal.

▶ **Theorem 2.** *For every $\epsilon > 0$, Split Contraction admits a factor $(2+\epsilon)$-FPT-approximation algorithm. In fact, for any $\epsilon > 0$, Split Contraction admits a $(2+\epsilon)$-approximate kernel with $\mathcal{O}(k^{f(\epsilon)})$ vertices.*

Given, Theorem 2, it is natural to ask whether Split Contraction admits a factor $(1+\epsilon)$-FPT-approximation algorithm, for every $\epsilon > 0$. We show that this is not true and obtain the following hardness result.

▶ **Theorem 3.** *Assuming Gap-ETH, no FPT time algorithm can approximate Split Contraction within a factor of $\left(\frac{5}{4} - \delta\right)$, for any fixed constant $\delta > 0$.*

Overview of the proofs of Theorems 2 and 3. Our proof for Theorem 2 uses ideas for $(1+\epsilon)$-approximate kernel for Clique Contraction (Theorem 1) and thus we omit its overview. Towards the proof of Theorem 3, we give a gap preserving reduction from a variant of the Densest-k-Subgraph problem (given a graph G and an integer k, find a subset $S \subseteq V(G)$ of k vertices that induces maximum number of edges). Chalermsook et al. [13] showed that, assuming Gap-ETH[1], for any $g = o(1)$, there is no FPT-time algorithm that, given an integer k and any graph G on n vertices that contains at least one k-clique, always output $S \subseteq V(G)$, of size k, such that $\mathsf{Den}(S) \geq k^{-g(k)}$. Here, $\mathsf{Den}(S) = |E(G[S])|/\binom{|S|}{2}$. We need a strengthening of this result that says that assuming Gap-ETH, for any $g = o(1)$ and for any constant $\alpha > 1$, there is no FPT-time algorithm that, given an integer k and any graph G on n vertices that contains at least one k-clique, always outputs $S \subseteq V(G)$, of size αk, such that $\mathsf{Den}(S) \geq k^{-g(k)}$. Starting from this result, we give a gap-preserving reduction to Split Contraction that takes FPT time and obtain Theorem 3. Given an instance (G, k) of Densest-k-Subgraph, we first use color coding to partition the edges into $t = \binom{k}{2}$ color classes such that every color class contains exactly one edge of a "densest subgraph" (or a clique). For each color class we make one *edge selection gadget*. Each edge selection gadget corresponding to the color class j consists of an independent set $\mathtt{ES}_j$ that contains a vertex corresponding to each edge in the color class$j$, and a *cap vertex* g_j that is adjacent to every vertex in $\mathtt{ES}_j$. Next, we add a sufficiently large clique Z of size $\rho \cdot |V(G)|$, where for every vertex $v \in V(G)$, we have ρ vertices. Every vertex in an edge selection gadget is adjacent to every vertex of Z, except those corresponding to the endpoints of the edge the vertex represents. Finally, we add a clique $\mathtt{SV}$ of size t that has one vertex s_j for each edge selection gadget. Make the vertex s_j adjacent to every vertex in $\mathtt{ES}_j$. We also add sufficient guards on vertices everywhere, so that "unwanted" contractions do not happen. The idea of the reduction is to contract edges in a way that the vertices in $\mathtt{SV}$, Z, and g_j, $j \in \{1, \ldots, t\}$, become a giant clique and other vertices become part of an independent set, resulting in a split graph. Towards this we first use $2t$ contractions so that g_j, s_j, and a vertex $a_j \in \mathtt{ES}_j$ are contracted into one. One way to ensure that they form a clique along with Z is to contract each of them to a vertex in Z. However, this will again require t edge contractions. We set our budget in a way that this is not possible. Thus, what we need is to destroy the non-neighbors of a_j. One way to do this again will be to match the vertices obtained after the first round of $2t$ contractions in a way that there are no non-adjacencies left. However, this will also cost $t/2$, and our budget does not allow this. The other option (which we take) is to take the union of all non-neighbors of a_j, say N, and contract each of them to one of the vertex in $Z \setminus N$. Observe that to minimize the contractions to get rid of non-neighbors of a_j, we would like to minimize $|N|$. This will happen when N spans a large number of edges. Thus, it precisely captures the Densest-k-Subgraph problem. The budget is chosen in a way that we get the desired gap-preserving reduction, which enables us to prove Theorem 3.

Our final result concerns Chordal Contraction. Lokshtanov et al. [38] showed that Chordal Contraction is W[2]-Hard. We observe that the existing W[2]-hardness reduction can be adapted to show the following theorem.

▶ **Theorem 4.** *Assuming* FPT$\neq$ *W[1], no* FPT *time algorithm can approximate* Chordal Contraction *within a factor of $F(k)$. Here, $F(k)$ is a function depending on k alone.*

[1] We refer the readers to [13] for the definition of Gap-ETH and related terms.

Overview of the proof of Theorem 4. Towards proving Theorem 4, we give a 1-approximate polynomial parameter transformation (1-appt) from Set Cover (given a universe U, a family of subsets $\mathcal{S}$, and an integer k, we shall decide the existence of a subfamily of size k that contains all the elements of U) to Chordal Contraction. That is, given any solution of size at most ℓ for Chordal Contraction, we can transform this into a solution for Set Cover of size at most ℓ. Karthik et al. [35] showed that assuming FPT$\neq$ W[1], no FPT time algorithm can approximate Set Cover within a factor of $F(k)$. Pipelining this result with our reduction we get Theorem 4.

Related Work. To the best of our knowledge, Heggernes et al. [33] was the first to explicitly study $\mathcal{F}$-Contraction from the viewpoint of Parameterized Complexity. They showed that in case $\mathcal{F}$ is the family of trees, $\mathcal{F}$-Contraction is FPT but does not admit a polynomial kernel, while in case $\mathcal{F}$ is the family of paths, the corresponding problem admits a faster algorithm and an $\mathcal{O}(k)$-vertex kernel. Golovach et al. [28] proved that if $\mathcal{F}$ is the family of planar graphs, then $\mathcal{F}$-Contraction is again FPT. Moreover, Cai and Guo [8] showed that in case $\mathcal{F}$ is the family of cliques, $\mathcal{F}$-Contraction is solvable in time $2^{\mathcal{O}(k \log k)} \cdot n^{\mathcal{O}(1)}$, while in case $\mathcal{F}$ is the family of chordal graphs, the problem is W[2]-Hard. Heggernes et al. [34] developed an FPT algorithm for the case where $\mathcal{F}$ is the family of bipartite graphs. Later, a faster algorithm was proposed by Guillemot and Marx [30].

Pioneering work of Lokshtanov et al. [40] on the approximate kernel is being followed by a series of papers generalizing/improving results mentioned in this work and establishing lossy kernels for various other problems. Lossy kernels for some variations of Connected Vertex Cover [21, 36], Connected Feedback Vertex Set [43], Steiner Tree [20] and Dominating Set [22, 44] have been established (also see [41, 45]). Krithika et al. [37] were first to study graph contraction problems from the lenses of lossy kernelization. They proved that for any $\alpha > 1$, Tree Contraction admits an α-lossy kernel with $\mathcal{O}(k^d)$ vertices, where $d = \lceil \alpha/(\alpha - 1) \rceil$. Agarwal et al. [1] proved similar result for $\mathcal{F}$-Contraction problems where graph class $\mathcal{F}$ is defined in parametric way from set of trees. Eiben et al. [21] obtained similar result for Connected $\mathcal{H}$-Hitting Set problem.

Organization. Due to space constraints, we omit some of the results from this extended abstract. We present the notations and preliminaries in Section 2 and in Section 3 we give the $(1 + \epsilon)$–approximate polynomial kernel for Clique Contraction.

2 Preliminaries

In this section, we give notations and definitions that we use throughout the paper. Unless specified, we will be using all general graph terminologies from the book of Diestel [15].

2.1 Graph Theoretic Definitions and Notations

For an undirected graph G, sets $V(G)$ and $E(G)$ denote the set of vertices and edges, respectively. Two vertices u, v in $V(G)$ are said to be *adjacent* if there is an edge uv in $E(G)$. The neighborhood of a vertex v, denoted by $N_G(v)$, is the set of vertices adjacent to v in G. For subset S of vertices, we define $N(S) = \bigcup_{v \in S} N(v)) \setminus S$. The subscript in the notation for the neighborhood is omitted if the graph under consideration is clear. For a set of edges F, set $V(F)$ denotes the endpoints of edges in F. For a subset S of $V(G)$, we denote the graph obtained by deleting S from G by $G - S$ and the subgraph of G induced on set S by $G[S]$. For two subsets S_1, S_2 of $V(G)$, we say S_1, S_2 *are adjacent if there exists an edge* with one endpoint in S_1 and other in S_2.

An edge e in G is a *chord* of a cycle C (resp. path P) if (i) both the endpoints of e are in C (resp. in P), and (ii) edge e is not in C (resp. not in P). An *induced cycle* (resp. *path*) is a cycle (resp. path) which has no chord. We denote induced cycle and path on ℓ vertices by C_ℓ and P_ℓ, respectively. A *complete graph* G is an undirected graph in which for every pair of vertices $u, v \in V(G)$, there is an edge uv in $E(G)$. As an immediate consequence of definition we get the following.

▶ **Lemma 5.** *A connected graph G is complete if and only if G does not contain an induced P_3.*

A *clique* is a subset of vertices in the graph that induces a complete graph. A set $I \subseteq V(G)$ of pairwise non-adjacent vertices is called an *independent set*. A graph G is a *split graph* if $V(G)$ can be partitioned into a clique and an independent set. For split graph G, partition (X, Y) is *split partition* if X is a clique and Y is an independent set. In this article, whenever we mention a split partition, we first mention the clique followed by the independent set. We will also use the following well-known characterization of split graphs. Let, $2K_2$ be a graph induced on four vertices, which contains exactly two edges and no isolated vertices.

▶ **Lemma 6** ([29])**.** *A graph G is a split graph if and only if it does not contain C_4, C_5 or $2K_2$ as an induced subgraph.*

A graph G is *chordal* if every induced cycle in G is a triangle; equivalently, if every cycle of length at least four has a chord. A vertex subset $S \subseteq V(G)$ is said to *cover* an edge $uv \in E(G)$ if $S \cap \{u, v\} \neq \emptyset$. A vertex subset $S \subseteq V(G)$ is called a *vertex cover* in G if it covers all the edges in G.

We start with the following observation, which is useful to find a large induced clique in the input graph. The *complement* of G, denoted by $\overline{G}$, is a graph whose vertex set is $V(G)$ and edge set is precisely those edges which are not present in $E(G)$. Note that given a graph G, if S is a set of vertices such that $G - S$ is a clique, then S is a vertex cover in the complement graphs of G, denoted by $\overline{G}$, as $\overline{G} - S$ is edgeless. Using the well-known factor 2-approximation algorithm for Vertex Cover [4], we have following.

▶ **Observation 2.1** ([4])**.** *There is a factor 2-approximation algorithm to compute a set of vertices whose deletion results in a complete graph.*

Using, Lemma 6 one can obtain a simple factor 5-approximation algorithm for deleting vertices to get a split graph.

▶ **Observation 2.2.** *There is a factor 5-approximation algorithm to compute a set of vertices whose deletion results in a split graph.*

Recently, for every $\epsilon > 0$, a factor $(2+\epsilon)$-approximation algorithm for deleting vertices to get a split graph has been obtained [39]. However, for our purposes Observation 2.2 will suffice.

2.2 Graph Contraction

The *contraction* of edge $e = uv$ in G deletes vertices u and v from G, and adds a new vertex, which is made adjacent to vertices that were adjacent to either u or v. Any parallel edges added in the process are deleted so that the graph remains simple. The resulting graph is denoted by G/e. Formally, for a given graph G and edge $e = uv$, we define G/e in the following way: $V(G/e) = (V(G) \cup \{w\}) \setminus \{u, v\}$ and $E(G/e) = \{xy \mid x, y \in V(G) \setminus \{u, v\}, xy \in E(G)\} \cup \{wx \mid x \in N_G(u) \cup N_G(v)\}$. For a subset of edges F in G, graph G/F denotes the graph obtained from G by repeatedly contracting edges in F until no such edge remains. We say that a graph G is *contractible* to a graph H if there exists an onto function $\psi : V(G) \to V(H)$ such that the following properties hold.

- For any vertex h in $V(H)$, graph $G[W(h)]$ is connected, where set $W(h) := \{v \in V(G) \mid \psi(v) = h\}$.
- For any two vertices h, h' in $V(H)$, edge hh' is present in H if and only if there exists an edge in G with one endpoint in $W(h)$ and another in $W(h')$.

For a vertex h in H, set $W(h)$ is called a *witness set* associated with h. We define *H-witness structure* of G, denoted by $\mathcal{W}$, as collection of all witness sets. Formally, $\mathcal{W} = \{W(h) \mid h \in V(H)\}$. Witness structure $\mathcal{W}$ is a partition of vertices in G, where each witness forms a connected set in G. Recall that if a *witness set* contains more than one vertex, then we call it *non-trivial* witness set, otherwise a *trivial* witness set.

If graph G has a H-witness structure, then graph H can be obtained from G by a series of edge contractions. For a fixed H-witness structure, let F be the union of spanning trees of all witness sets. By convention, the spanning tree of a singleton set is an empty set. Thus, to obtain H from G, it is *sufficient* to contract edges in F. If such witness structure exists, then we say that graph G is contractible to H. We say that graph G is *k-contractible* to H if cardinality of F is at most k. In other words, H can be obtained from G by at most k edge contractions. Following observation is an immediate consequence of definitions.

▶ **Observation 2.3.** *If graph G is k-contractible to graph H, then the following statements are true.*

- *For any witness set W in a H-witness structure of G, the cardinality of W is at most $k + 1$.*
- *For a fixed H-witness structure, the number of vertices in G, which are contained in non-trivial witness sets is at most $2k$.*

In the following two observations, we state that if a graph can be transformed into a clique or a split graph by contracting few edges, then it can also be converted into a clique or split graph by deleting few vertices.

▶ **Observation 2.4.** *If a graph G is k-contractible to a clique, then G can be converted into a clique by deleting at most $2k$ vertices.*

Proof. Let F be a set of edges of size at most k such that G/F is a clique. Let $\mathcal{W}$ be a G/F-witness structure of G. Let X be a set of all vertices which are contained in the non-trivial witness sets in $\mathcal{W}$. By Observation 2.3, size of X is at most $2k$. Any two vertices in $V(G) \setminus X$ are adjacent to each other as these vertices form singleton sets, which are adjacent in G/F. Hence, G can be converted into a clique by deleting vertices in X. ◀

▶ **Observation 2.5.** *If a graph G is k-contractible to a split graph then G can be converted into a split graph by deleting at most $2k$ vertices.*

Proof. For graph G, let F be the set of edges such that G/F is a split graph and $|F| \leq k$. Let $V(F)$ be the collection of all endpoints of edges in F. Since cardinality of F is at most k, $|V(F)|$ is at most $2k$. We argue that $G - V(F)$ is a split graph. For the sake of contradiction, assume that $G - V(F)$ is not a split graph. We know that a graph is split if and only if it does not contain induced C_4, C_5 or $2K_2$. This implies that there exists a set of vertices V' in $V(G) \setminus V(F)$ such that $G[V']$ is either C_4, C_5 or $2K_2$. Since no edge in F is incident on any vertices in V', $G/F[V']$ is isomorphic to $G[V']$. Hence, there exists a C_4, C_5 or $2K_2$ in G/F contradicting the fact that G/F is a split graph. Hence, our assumption is wrong and $G - V(F)$ is a split graph. ◀

Consider a connected graph G which is k-contractible to the clique K_ℓ. Let $\mathcal{W}$ be a K_ℓ-witness structure of G. The following observation gives a sufficient condition for obtaining a witness structure of an induced subgraph of G from $\mathcal{W}$.

▶ **Observation 2.6.** *Let $\mathcal{W}$ be a clique witness structure of G. If there exists two different witness sets $W(t_1), W(t_2)$ in $\mathcal{W}$ and a vertex v in $W(t_1)$ such that the set $W(t) = (W(t_1) \cup W(t_2)) \setminus \{v\}$ is a connected set in $G - \{v\}$, then $\mathcal{W}'$ is a clique witness structure of $G - \{v\}$, where $\mathcal{W}'$ is obtained from $\mathcal{W}$ by removing $W(t_1), W(t_2)$ and adding $W(t)$.*

Proof. Let $G' = G - \{v\}$. Note that $\mathcal{W}'$ is a partition of vertices in G'. Any set in $\mathcal{W}' \setminus \{W(t)\}$ is a witness set in $\mathcal{W}$ and does not contain v. Hence, these sets are connected in G'. Since $G'[W(t)]$ is also connected, all the witness sets in $\mathcal{W}'$ are connected in G'.

Consider any two witness sets $W(t'), W(t'')$ in $\mathcal{W}'$. If none of these two is equal to $W(t)$ then both of these sets are present in $\mathcal{W}$. Since none of these witness sets contains vertex v, they are adjacent to each other in G'. Now, consider a case when one of them, say $W(t'')$, is equal to $W(t)$. As witness sets $W(t')$ and $W(t_2)$ are present in $\mathcal{W}$, there exists an edge with one endpoint in $W(t')$ and another in $W(t_2)$. The same edge is present in graph G' as it is not incident on v. Since $W(t_2)$ is subset of $W(t)$, sets $W(t')$ and $W(t)$ are adjacent in G'. Hence any two witness sets in $\mathcal{W}'$ are adjacent to each other. This implies that $\mathcal{W}'$ is a clique witness structure of graph $G - \{v\}$. ◀

In the case of Split Contraction, the following observation guarantees the existence of witness structure with a particular property.

▶ **Observation 2.7.** *For a connected graph G, let F be a set of edges such that G/F is a split graph. Then, there exists a set of edges F' which satisfy the following properties: (i) G/F' is a split graph. (ii) The number of edges in F' is at most $|F|$. (iii) There exists a split partition of G/F' such that all vertices in G/F' which correspond to a non-trivial witness set in G/F'-witness structure of G are in clique side.*

Proof. Let (C, I) be a split partition of vertices of G/F such that C is a clique and I is an independent set. If all the vertices corresponding to non-trivial witness sets are in C, then the observation is true. Consider a vertex a in I which corresponds to a non-trivial witness set W_a. Since G is connected, G/F is a connected split graph. This implies that there exists a vertex, say b, in C which is adjacent to a in G/F. We denote witness set corresponding to b by W_b. We construct a new witness structure by shifting all but one vertices in W_a to W_b. Since ab is an edge in G/F, there exists an edge in G with one endpoint in W_a and another in W_b. Let that edge be $u_a u_b$ with vertices u_a and u_b contained in sets W_a and W_b, respectively. Consider a spanning tree T of graph $G[W_a]$ which is rooted at u_a. We can replace edges in F whose both endpoints are in $V(W_a)$ with $E(T)$ to obtain another set of edges F^* such that G/F^* is a split graph. Formally, $F^* = (F \cup E(T)) \setminus (E(G[W_a]) \cap F)$. Note that the number of edges in F^* and F are same. Let v_1 be a leaf vertex in T and v_2 be its unique neighbor. Consider $F' = (F^* \cup \{u_a u_b\}) \setminus \{v_1 v_2\}$. Since edge $v_1 v_2$ is in F^* and $u_a u_b$ is not in F^*, $|F'| = |F^*|$. We now argue that G/F' is also a split graph. Let $\mathcal{W}'$ be the G/F'-witness structure of G. Note that $\mathcal{W}'$ can be obtained from G/F^*-witness structure $\mathcal{W}^*$ of G by replacing W_a by $\{v_1\}$ and W_b by $W_b \cup (W_a \setminus \{v_1\})$. Since all other witness set remains unchanged any witness set which was adjacent to W_b is also adjacent to $W_b \cup (W_a \setminus \{v_1\})$. Similarly, any witness set which was not adjacent to W_a is not adjacent to $\{v_2\}$. In other words, this operation of shifting edges did not remove any vertex from the neighborhood of b (which is in C) nor it added any vertex in the neighborhood of a (which is in I). Hence, G/F' is also a split graph with (C, I) as one of its split partition. Note that there exists a split partition of G/F' such that the number of vertices in the independent side corresponding to non-trivial witness set is one less than the number of vertices in I which corresponds to non-trivial witness sets. Hence, by repeating this process at most $|V(G)|$ times, we get a set of edges that satisfy three properties mentioned in the observation. ◀

2.3 Parameterized Complexity and Lossy Kernelization

An important notion in parameterized complexity is *kernelization*, which captures the efficiency of data reduction techniques. A parameterized problem Π admits a kernel of size $g(k)$ (or $g(k)$-kernel) if there is a polynomial time algorithm (called *kernelization algorithm*) which takes as input (I, k), and returns an instance (I', k') of Π such that: (i) (I, k) is a yes-instance if and only if (I', k') is a yes-instance; and (ii) $|I'| + k' \leq g(k)$, where $g(\cdot)$ is a computable function whose value depends only on k. Depending on whether the function $g(\cdot)$ is *linear, polynomial* or *exponential*, the problem is said to admit a *linear, polynomial* or *exponential kernel*, respectively. We refer to the corresponding chapters in the books [25, 14, 16, 23, 42] for a detailed introduction to the field of kernelization.

In lossy kernelization, we work with the optimization analog of parameterized problem. Along with an instance and a parameter, an optimization analog of the problem also has a string called *solution*. We start with the definition of a *parameterized optimization problem*. It is the parameterized analog of an optimization problem used in the theory of approximation algorithms.

▶ **Definition 7** (Parameterized Optimization Problem)**.** *A parameterized optimization problem is a computable function* $\Pi : \Sigma^* \times \mathbb{N} \times \Sigma^* \mapsto \mathbb{R} \cup \{\pm\infty\}$*. The instances of* Π *are pairs* $(I, k) \in \Sigma^* \times \mathbb{N}$ *and a solution to* (I, k) *which is simply a string* $S \in \Sigma^*$ *such that* $|S| \leq |I| + k$*.*

The *value* of a solution S is $\Pi(I, k, S)$. In this paper, all optimization problems are minimization problems. Therefore, we present the rest of the section only with respect to parameterized *minimization* problem. The *optimum value* of (I, k) is defined as:

$$\mathrm{OPT}_\Pi(I, k) = \min_{S \in \Sigma^*,\, |S| \leq |I|+k} \Pi(I, k, S),$$

and an *optimum solution* for (I, k) is a solution S such that $\Pi(I, k, S) = \mathrm{OPT}_\Pi(I, k)$. For a constant $c > 1$, S is *c-factor approximate* solution for (I, k) if $\frac{\Pi(I,k,S)}{OPT_\Pi(I,k)} \leq c$. We omit the subscript Π in the notation for optimum value if the problem under consideration is clear from the context.

For some parameterized optimization problems we are unable to obtain FPT algorithms, and we are also unable to find satisfactory polynomial time approximation algorithms. In this case one might aim for FPTapproximation algorithms, algorithms that run in time $f(k)n^c$ and provide good approximate solutions to the instance.

▶ **Definition 8.** *Let* $\alpha \geq 1$ *be constant. A fixed parameter tractable* α*-approximation algorithm for a parameterized optimization problem* Π *is an algorithm that takes as input an instance* (I, k)*, runs in time* $f(k)|I|^{\mathcal{O}(1)}$*, and outputs a solution* S *such that* $\Pi(I, k, S) \leq \alpha \cdot OPT(I, k)$ *if* Π *is a minimization problem, and* $\alpha \cdot \Pi(I, k, S) \geq OPT(I, k)$ *if* Π *is a maximization problem.*

Note that Definition 8 only defines constant factor FPT-approximation algorithms. The definition can in a natural way be extended to approximation algorithms whose approximation ratio depends on the parameter k, on the instance I, or on both. Next, we define an α-*approximate polynomial-time preprocessing algorithm* for a parameterized minimization problem Π as follows.

▶ **Definition 9** (α-Approximate Polynomial-time Preprocessing Algorithm)**.** *Let* $\alpha \geq 1$ *be a real number and* Π *be a parameterized minimization problem. An* α*-approximate polynomial-time preprocessing algorithm is defined as a pair of polynomial-time algorithms, called the* reduction algorithm *and the* solution lifting algorithm*, that satisfy the following properties.*

- *Given an instance* (I, k) *of* Π*, the reduction algorithm computes an instance* (I', k') *of* Π.
- *Given instances* (I, k) *and* (I', k') *of* Π*, and a solution* S' *to* (I', k')*, the solution lifting algorithm computes a solution* S *to* (I, k) *such that* $\frac{\Pi(I,k,S)}{OPT(I,k)} \leq \alpha \cdot \frac{\Pi(I',k',S')}{OPT(I',k')}$.

We sometimes refer α-approximate polynomial-time preprocessing algorithm kernel as *α-lossy rule* or *α-reduction rule*.

3 Lossy Kernel for Clique Contraction

In this section, we present a lossy kernel for Clique Contraction. We first define a natural optimization version of the problem.

$$\textsc{Clc}(G, k, F) = \begin{cases} \min\{|F|, k+1\} & \text{if } G/F \text{ is a clique} \\ \infty & \text{otherwise} \end{cases}$$

If the number of vertices in the input graph is at most $k+3$, then we can return the same instance as a kernel for the given problem. Further, we assume that the input graph is connected; otherwise, it can not be edited into a clique by edge contraction only. Thus, we only consider *connected graphs with at least* $k+3$ *vertices*. By the definition of optimization problem, for any set of edges F, if G/F is a clique, then the maximum value of $\textsc{Clc}(G, k, F)$ is $k+1$. Hence, any spanning tree of G is a solution of cost $k+1$. We call it a *trivial solution* for the given instance. Consider an instance $(P_4, 1)$, where P_4 is a path on four vertices. One needs to contract at least two edges to convert P_4 into a clique. We call $(P_4, 1)$ a *trivial No-instance* for this problem. Finally, we assume that we are given an $\epsilon > 0$.

We start with a reduction rule, which says that if the minimum number of vertices that need to be deleted from an input graph to obtain a clique is large, then we can return a trivial instance as a lossy kernel.

▶ **Reduction Rule 3.1.** *For a given instance* (G, k)*, apply the algorithm mentioned in Observation 2.1 to find a set* X *such that* $G - X$ *is a clique. If the size of* X *is greater than* $4k$*, then return* $(P_4, 1)$.

▶ **Lemma 10.** *Reduction Rule 3.1 is a* 1*-reduction rule.*

Proof. Let (G, k) be an instance of Clique Contraction such that the Reduction Rule 3.1 returns $(P_4, 1)$ when applied on it. The solution lifting algorithm returns a spanning tree F of G. Note that for a set of edges F', if P_4/F' is a clique then F' contains at least two edges. This implies $\textsc{Clc}(P_4, 1, F') = 2$ and $\text{OPT}(P_4, 1) = 2$.

Since a factor 2-approximation algorithm returned a set of size strictly more than $4k$, for any set X' of size at most $2k$, $G - X'$ is not a clique. But by Observation 2.4, if G is k-contractible to a clique then G can be edited into a clique by deleting at most $2k$ vertices. Hence, for any set of edges F^* if G/F^* is a clique, then the size of F^* is at least $k+1$. This implies that $\text{OPT}(G, k) = k+1$, and for a spanning tree F of G, $\textsc{Clc}(G, k, F) = k+1$.

Combining these values, we get $\frac{\textsc{Clc}(G,k,F)}{\text{OPT}(G,k)} = \frac{k+1}{k+1} = \frac{2}{2} = \frac{\textsc{Clc}(P_4,1,F')}{\text{OPT}(P_4,1)}$. This implies that if F' is factor c-approximate solution for $(P_4, 1)$, then F is factor c-approximate solution for (G, k). This concludes the proof. ◀

We now consider an instance (G, k) for which Reduction Rule 3.1 does not return a trivial instance. This implies that for a given graph G, in polynomial time, one can find a partition (X, Y) of $V(G)$ such that $G - X = G[Y]$ is a clique and $|X|$ is at most $4k$. For $\epsilon > 0$, find a smallest integer d, such that $\frac{d+1}{d} \leq 1 + \epsilon$. In other words, fix $d = \lceil \frac{1}{\epsilon} \rceil$. We note that if the

number of vertices in the graph is at most $\mathcal{O}(k^{d+1})$, then the algorithm returns this graph as a lossy kernel of the desired size. Hence, without loss of generality, we assume that the number of vertices in the graph is larger than $\mathcal{O}(k^{d+1})$.

Given an instance (G, k), a partition (X, Y) of $V(G)$ with $G[Y]$ being a clique, and an integer d, consider the following two marking schemes.

▶ **Marking Scheme 3.1.** *For a subset A of X, let $M_1(A)$ be the set of vertices in Y whose neighborhood contains A. For every subset A of X which is of size at most d, mark a vertex in $M_1(A)$.*

Formally, $M_1(A) = \{y \in Y | A \subseteq N(y)\}$. If $M_1(A)$ is an empty set, then the marking scheme does not mark any vertex. If it is non-empty, then the marking scheme arbitrarily chooses a vertex and marks it.

▶ **Marking Scheme 3.2.** *For a subset A of X, let $M_2(A)$ be the set of vertices in Y whose neighborhood does not intersect A. For every subset A of X which is of size at most d, mark $2k + 1$ vertices in $M_2(A)$.*

Formally, $M_2(A) = \{y \in Y | N(y) \cap A = \emptyset\}$. If the number of vertices in $M_2(A)$ is at most $2k + 1$, then the marking scheme marks all vertices in $M_2(A)$. If it is larger than $2k + 1$, then it arbitrarily chooses $2k + 1$ vertices and marks them.

▶ **Reduction Rule 3.2.** *For a given instance (G, k), partition (X, Y) of $V(G)$ with $G[Y]$ being a clique, and an integer d, apply the Marking Schemes 3.1 and 3.2. Let G' be the graph obtained from G by deleting all the unmarked vertices in Y. Return the instance (G', k).*

Above reduction rule can be applied in time $|X|^d \cdot |V(G)|^{\mathcal{O}(1)} = \mathcal{O}(k^{\mathcal{O}(d)}|V(G)|^{\mathcal{O}(1)})$ as $|X|$ is at most $4k$. Note that G' is an induced subgraph of G. We first show that since G is a connected graph, G' is also connected. In the following lemma, we prove a stronger statement.

▶ **Lemma 11.** *Consider instance (G, k) of Clique Contraction. Let Y' be the set of vertices marked by Marking Scheme 3.1 or 3.2 for some positive integer d. For any subset Y'' of $Y \setminus Y'$, let G'' be the graph obtained from G by deleting Y''. Then, G'' is connected.*

Proof. Recall that, by our assumption, G is connected and Y is a clique in G. Hence, for every vertex in X, there exists a path from it to some vertex in Y. By the construction of G'', $(X, Y \setminus Y'')$ forms a partition of $V(G'')$ and $Y \setminus Y''$ is a clique in G''. To prove that G'' is connected, it is sufficient to prove that for every vertex in X, there exists a path from it to a vertex in $Y \setminus Y''$ in G.

Fix an arbitrary vertex, say x, in X. Consider a path P from x to y in G, where y is some vertex in Y. Without loss of generality, we can assume that y is the only vertex in $V(P) \cap Y$. We argue that there exists another path, say P_1, from x to a vertex in $Y \setminus Y''$. If y is in $Y \setminus Y''$ then $P_1 = P$ is a desired path. Consider the case when y is in Y''. Let x_0 be the vertex in $V(P)$ which is adjacent with y. Note that x_0 may be same as x. As Marking Scheme 3.1 considers all subsets of size at most d, it considered singleton set $\{x_0\}$. As x_0 is adjacent with y, we have $\{x_0\} \subseteq N(y)$. Since y is in Y'', and hence unmarked, there exists a vertex, say y_1, in Y which has been marked by Marking Scheme 3.1. Consider a path P_1 obtained from P by deleting vertex y (and hence edge x_0y) and adding vertex y_1 with edge x_0y_1. This is a desired path from x to a vertex in $Y \setminus Y''$. As x is an arbitrary vertex in X, this statement is true for any vertex in X and hence G'' is connected. ◀

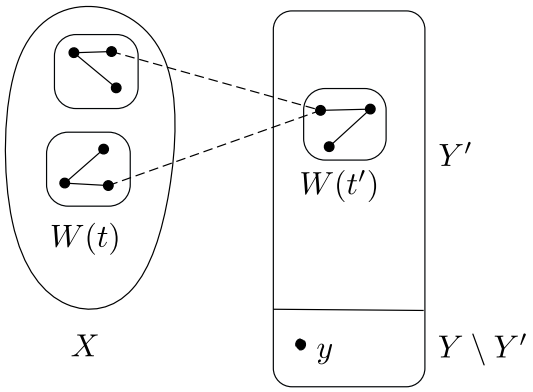


Figure 1 Straight lines (e.g within $W(t)$) represent edges in original solution F. Dashed lines (e.g. across $W(t)$ and $W(t')$) represents extra edges added to solution F. Please refer to the proof of Lemma 12.

Thus, because of Lemma 11, from now onwards, we assume that G' is connected. In fact, in our one of the proof, we will iteratively remove vertices from $Y \setminus Y'$, and Lemma 11 ensures that the graph at each step remains connected. In the following lemma, we argue that given a solution for (G', k), we can construct a solution of almost the same size for (G, k).

▶ **Lemma 12.** *Let (G', k) be the instance returned by Reduction Rule 3.2 when applied on an instance (G, k). If there exists a set of edges of size at most k, say F', such that G'/F' is a clique, then there exists a set of edges F such that G/F is a clique and cardinality of F is at most $(1 + \epsilon) \cdot |F'|$.*

Proof. If no vertex in Y is deleted, then G' and G are identical graphs, and the statement is true. We assume that at least one vertex in Y is deleted. Let Y' be the set of vertices in Y, which are marked. Note that the sets X, Y' forms a partition of $V(G')$ such that Y' is a clique and a proper subset of Y. Let $\mathcal{W}'$ be a G'/F'-witness structure of G'. We construct a clique witness structure $\mathcal{W}$ of G from $\mathcal{W}'$ by adding singleton witness sets $\{y\}$ for every vertex y in $Y \setminus Y'$. Since $G[Y \setminus Y']$ is a clique in G, any two newly added witness sets are adjacent to each other. Moreover, any witness set in $\mathcal{W}'$, which intersects Y' is also adjacent to the newly added witness sets. We now consider witness sets in $\mathcal{W}'$, which do not intersect Y'.

Let $\mathcal{W}^\star$ be a collection of witness sets $W(t)$ in $\mathcal{W}'$ such that $W(t)$ is contained in X and there exists a vertex y in $Y \setminus Y'$ whose neighborhood does not intersect with $W(t)$. See Figure 1. We argue that every witness set in $\mathcal{W}^\star$ has at least $d + 1$ vertices. For the sake of contradiction, assume that there exists a witness set $W(t)$ in $\mathcal{W}^\star$ which contains at most d vertices. Since Marking Scheme 3.2 iterated over all the subsets of X of size at most d, it also considered $W(t)$ while marking. Note that the vertex y belongs to the set $M_2(W(t))$. Since y is unmarked, there are $2k + 1$ vertices in $M_2(W(t))$ which have been marked. All these marked vertices are in G'. Since the cardinality of F' is at most k, the number of vertices in $V(F')$ is at most $2k$. Hence, at least one marked vertex in $M_2(W(t))$ is a singleton witness set in $\mathcal{W}'$. However, there is no edge between this singleton witness set and $W(t)$. This non-existence of an edge contradicts the fact that any two witness sets in $\mathcal{W}'$ are adjacent to each other in G'. Hence, our assumption is wrong, and $W(t)$ has at least $d + 1$ vertices.

Next, we show that there exists a witness set in $\mathcal{W}'$ that intersects Y'. This is ensured by the fact that G' is connected, and we are in the case where at least one vertex in Y is deleted. The last assertion implies that Y' is non-empty, and hence there must be a witness set in $\mathcal{W}'$ that intersects Y'. Let $W(t')$ be a witness set in $\mathcal{W}'$ that intersects Y'. Note that $W(t')$ is adjacent to every vertex in $Y \setminus Y'$. Let $W(t)$ be a witness set in $\mathcal{W}^\star$. Since $W(t')$ and $W(t)$ are two witness sets in the G'/F'-witness structure, there exists an edge with one endpoint in $W(t')$ and another in $W(t)$. Therefore, the set $W(t') \cup W(t)$ is adjacent to every other witness set in $\mathcal{W}$.

We now describe how to obtain F from F'. We initialize $F = F'$. For every witness set $W(t)$ in $\mathcal{W}^\star$ add an edge between $W(t)$ and $W(t')$ to the set F'. Equivalently, we construct a new witness set by taking the union of $W(t')$ and all witness sets $W(t)$ in $\mathcal{W}^\star$. This witness set is adjacent to every vertex in $Y \setminus Y'$, and hence G/F is a clique. We now argue the size bound on F. Note that we have added one extra edge for every witness set $W(t)$ in $\mathcal{W}^\star$. We also know that every such witness set has at least $d+1$ vertices. Hence, we have added one extra edge for at least d edges in the solution F'. Moreover, since witness sets in $\mathcal{W}^\star$ are vertex disjoint, no edge in F can be part of two witness sets. This implies that the number of edges in F' is at most $({}^{d+1}/_{d})|F| \leq (1+\epsilon) \cdot |F|$. ◀

In the following lemma, we argue that the value of the optimum solution for the reduced instance can be upper bounded by the value of an optimum solution for the original instance.

▶ **Lemma 13.** *Let (G', k) be the instance returned by Reduction Rule 3.2 when applied on an instance (G, k). If $OPT(G, k) \leq k$, then $OPT(G', k) \leq OPT(G, k)$.*

Proof. Let F be a set of at most k edges in G such that $\mathrm{OPT}(G, k) = \textsc{Clc}(G, k, F)$ and $\mathcal{W}$ be a G/F-witness structure of G. Since we are working with a minimization problem, to prove this lemma it is sufficient to find a solution for G' which is of size $|F|$. Recall that (X, Y) is a partition of $V(G)$ such that $G - X = G[Y]$ is a clique. Let Y' be the set of vertices that were marked by either of the marking schemes. In other words, (X, Y') is a partition of G' such that $G' - X = G'[Y]$ is a clique. We proceed as follows. At each step, we construct graph $G^\star$ from G by deleting one or more vertices of $Y \setminus Y'$. Simultaneously, we also construct a set of edges $F^\star$ from F by either replacing the existing edges by new ones or by simply adding extra edges to F. At any intermediate state, we ensure that $G^\star/F^\star$ is a clique, and the number of edges in $F^\star$ is at most $|F|$. Let $F^\circ = F$ be an optimum solution for the input instance (G, k). *For notational convenience, we rename $G^\star$ to G and $F^\star$ to F at regular intervals but do not change F°.*

To obtain $G^\star$ and $F^\star$, we delete witness sets which are subsets of $Y \setminus Y'$ (Condition 3.1) and modify the ones which intersect with $Y \setminus Y'$. Every witness set of latter type intersects with Y' or X or both. We partition these non-trivial witness sets in $\mathcal{W}$ into two groups depending on whether the intersection with X is empty (Condition 3.2) or not (Condition 3.3). We first modify the witness sets that satisfy the least indexed condition. If there does not exist a witness set which satisfies either of these three conditions, then $Y \setminus Y'$ is an empty set, and the lemma is vacuously true.

▶ **Condition 3.1.** *There exists a witness set $W(t)$ in $\mathcal{W}$ which is a subset of $Y \setminus Y'$.*

Construct $G^\star$ from G by deleting the witness sets $W(t)$ in $\mathcal{W}$. Let $F^\star$ be obtained from F by deleting those edges whose both the endpoints are in $W(t)$. Since the class of cliques is closed under vertex deletion, $G^\star/F^\star$ is a clique, and as we only deleted edges from F, we have $|F^*| \leq |F|$. We repeat this process until there exists a witness set that satisfies Condition 3.1.

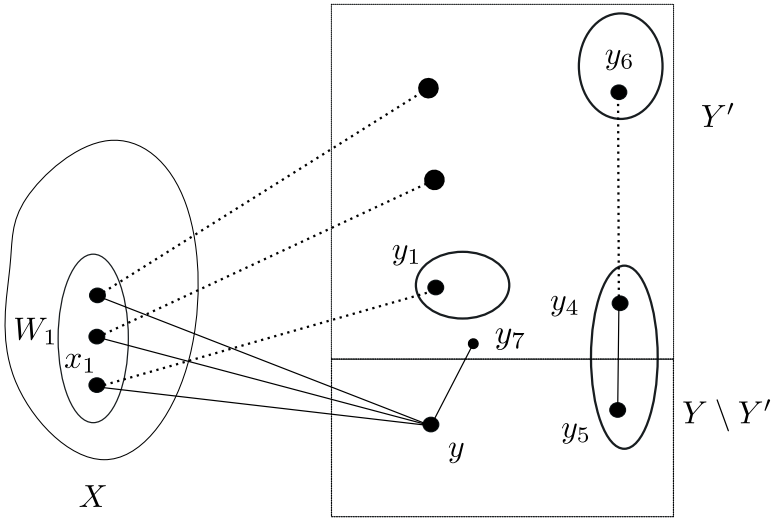


Figure 2 Straight lines (e.g. y_4y_5) represent edges in original solution F. Dotted lines (e.g. y_4y_6) represents edges which are replaced for some edges in F. Please refer to the proof of Lemma 13.

At this stage we rename $G^\star$ to G and $F^\star$ to F.

▶ **Condition 3.2.** *There exists a witness set $W(t)$ in $\mathcal{W}$ which contains vertices from $Y \setminus Y'$ but does not intersect X.*

Since $W(t)$ is not contained in $Y \setminus Y'$ and $W(t) \cap X$ is empty it must intersect with Y'. See Figure 2. Let y_4 and y_5 be vertices in $W(t) \cap Y'$ and $W(t) \cap (Y \setminus Y')$, respectively. Let $W(t_1)$, different from $W(t)$, be a witness set which intersects Y'. Since Y' is large and non-empty, such a witness set exists. Let y_6 be a vertex in the set $W(t_1) \cap Y'$. Consider the witness sets $W(t), W(t_1)$ and vertex y_5 in $W(t)$ in graph G. Lemma 11 implies that these witness sets satisfy the premise of Observation 2.6. This implies $\mathcal{W}^\star$ is a clique witness structure of $G - \{y_5\}$, where $\mathcal{W}^\star$ is obtained from $\mathcal{W}$ by removing $W(t), W(t_1)$ and adding $(W(t) \cup W(t_1)) \setminus \{y_5\}$. This corresponds to replacing an edge in F which was incident to y_5 with the one across $W(t)$ and $W(t_1)$. For example, in Figure 2, we replace edge y_4y_5 in the set F with an edge y_4y_6 to obtain a solution for $G - \{y_5\}$. An edge in F has been replaced with another edge and one vertex in $Y \setminus Y'$ is deleted. The size of $F^\star$ is same as that of F and $G^\star/F^\star$ is a clique. We repeat this process until there exist a witness set which satisfies Condition 3.2.

At this stage we rename $G^\star$ to G and $F^\star$ to F.

▶ **Condition 3.3.** *There exists a witness set $W(t)$ in $\mathcal{W}$ which contains vertices from $Y \setminus Y'$ and intersects X.*

Let y be a vertex in $W(t) \cap (Y \setminus Y')$, X_t be the set of vertices in $W(t) \cap X$ which are adjacent to y via edges in F, and Q_t be the set of vertices in $W(t) \cap Y$ which are adjacent to y via edges in F. We find a *substitute* for y in Y'. If the set X_t is empty then the vertex y is adjacent only with the vertices of Y, in this case the edges incident to y can be replaced as

mentioned in the Condition 3.2. Assume that X_t is non-empty. For every vertex x in X_t the set $\{x\}$ is considered by Marking Scheme 3.1. Since y is adjacent to every vertex x in X_t, the set $M_1(\{x\})$ is non-empty. As y is in $Y \setminus Y'$, and hence unmarked, for every x in X_t, there is a vertex in $M_1(\{x\})$, say y_x, different from y which has been marked. We construct $F^\star$ from F by the following operation: For every vertex x in X_t, replace the edge xy in F by xy_x. Fix a vertex x_o in X_t, and for every vertex u in Q_t, replace the edge uy in F with uy_{x_o}. Since we are replacing a set of edges in F with another set of edges of same size we have $|F^\star| \leq |F|$. (For example, in Figure 2, $X_t = W_1$ and $Q_t = \{y_7\}$. Edges yx_1, yy_7 are replaced by x_1y_1, y_1y_7 resp.) We argue that if $G^\star$ is obtained from G by removing y, then $G^\star/F^\star$ is a clique.

We argue that contracting edges in $F^\star$ partitions $W(t)$ into $|X_t| + |Q_t|$ many parts and merges each part with some witness set in $\mathcal{W} \setminus \{W(t)\}$. Recall that F contains a spanning tree of graph $G[W(t)]$. Let T be a spanning tree of $G[W(t)]$ such that $E(T) \subseteq F$ and T contains all edges in F that are incident on y. It is easy to see that such a spanning tree exists. Let y be the root of tree T. For every z in $X_t \cup Q_t$, let $W'(z)$ be the set of vertices in the subtree of T rooted at z. As $V(T) = W(t)$, set $\{W'(z) |\ z \in X_t \cup Q_t\}$ is a partition of $W(t) \setminus \{y\}$. For every x in X_t, let $W(y_x)$ be the witness set in $\mathcal{W}$ containing the vertex y_x. For every x in $X_t \setminus \{x_o\}$, let $W^\star(y_x)$ be the set $W(y_x) \cup W'(x)$. Let $W^\star(y_{x_o})$ be the set $W(y_{x_o}) \cup W'(x_o) \cup \bigcup_{y'} W'(y')$ for all y' in Q_t. We obtain $\mathcal{W}^\star$ from $\mathcal{W}$ by removing $W(t)$ and $W(y_x)$ for every x in X_t, and adding the sets $W^\star(y_x)$ for every x in X_t. Since $W^\star(y_x)$ contains the set $W(y_x)$ which was adjacent to every witness set in $\mathcal{W}$, $W^\star(y_x)$ will be adjacent with every witness set in $\mathcal{W}^\star$. We repeat this process until there exists a witness set that satisfies this condition.

Any vertex in $Y \setminus Y'$ must be a part of some witness set in $\mathcal{W}$, and any witness set in $\mathcal{W}$ satisfies at least one of the above conditions. If there are no witness sets that satisfy these conditions, then $Y \setminus Y'$ is empty. This implies $G^\star = G'$ and there exists a solution $F^\star$ of size at most $|F^\circ|$. This concludes the proof of the lemma. ◀

We are now in a position to prove the following lemma.

▶ **Lemma 14.** *Reduction Rule 3.2, along with a solution lifting algorithm, is an* $(1+\epsilon)$*-reduction rule.*

Proof. Let (G', k) be the instance returned by Reduction Rule 3.2 when applied on an instance (G, k). We present a solution lifting algorithm as follows. For a solution F' for (G, k) if $\textsc{Clc}(G', k, F') = k+1$, then the solution lifting algorithm returns a spanning tree F of G (a trivial solution) as solution for (G, k). In this case, $\textsc{Clc}(G, k, F) = \textsc{Clc}(G', k, F')$. If $\textsc{Clc}(G', k, F') \leq k$, then size of F' is at most k and G'/F' is a clique. Solution lifting algorithm uses Lemma 12 to construct a solution F for (G, k) such that cardinality of F is at most $(1+\epsilon) \cdot |F'|$. In this case, $\textsc{Clc}(G, k, F) \leq (1+\epsilon) \cdot \textsc{Clc}(G', k, F')$. Hence, there exists a solution lifting algorithm which given a solution F' for (G', k') returns a solution F for (G, k) such that $\textsc{Clc}(G, k, F) \leq (1+\epsilon) \cdot \textsc{Clc}(G', k, F')$.

If $\mathrm{OPT}(G, k) \leq k$, then by Lemma 13, $\mathrm{OPT}(G', k) \leq \mathrm{OPT}(G, k)$. If $\mathrm{OPT}(G, k) = k+1$ then $\mathrm{OPT}(G', k) \leq k+1 = \mathrm{OPT}(G, k)$. Hence in either case, $\mathrm{OPT}(G', k) \leq \mathrm{OPT}(G, k)$.

Combining the two inequalities, we get $\frac{\textsc{Clc}(G,k,F)}{\mathrm{OPT}(G,k)} \leq \frac{(1+\epsilon)\cdot\textsc{Clc}(G',k,F')}{\mathrm{OPT}(G',k)}$. This implies that if F' is a factor c-approximate solution for (G', k) then F is a factor $(c \cdot (1+\epsilon))$-approximate solution for (G, k). This concludes the proof. ◀

We are now in a position to present the main result of this section.

Proof. (of Theorem 1) For a given instance (G, k) with $|V(G)| \geq k + 3$, a kernelization algorithm applies the Reduction Rule 3.1. If it returns a trivial instance, then the statement is vacuously true. If it does not return a trivial instance, then the algorithm partitions $V(G)$ into two sets (X, Y) such that $G - X = G[Y]$ is a clique and size of X is at most $4k$. Then the algorithm applies the Reduction Rule 3.2 on the instance (G, k) with the partition (X, Y) and the integer $d = \lceil \frac{1}{\epsilon} \rceil$. The algorithm returns the reduced instance as $(1 + \epsilon)$-lossy kernel for (G, k).

The correctness of the algorithm follows from Lemma 10 and Lemma 14 combined with the fact that Reduction Rule 3.2 is applied at most once. By Observation 2.1, Reduction Rule 3.1 can be applied in polynomial time. The size of the instance returned by Reduction Rule 3.2 is at most $\mathcal{O}((4k)^d \cdot (2k + 1) + 4k) = \mathcal{O}(k^{d+1})$. Reduction Rule 3.2 can be applied in time $n^{\mathcal{O}(1)}$ if the number of vertices in (G, k) is more than $\mathcal{O}(k^{d+1})$. ◀

References

1. Akanksha Agarwal, Saket Saurabh, and Prafullkumar Tale. On the parameterized complexity of contraction to generalization of trees. *Theory of Computing Systems*, pages 1–28, 2017.
2. Akanksha Agrawal, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. Split contraction: The untold story. In *STACS*, pages 5:1–5:14, 2017. doi:10.4230/LIPIcs.STACS.2017.5.
3. Takao Asano and Tomio Hirata. Edge-contraction problems. *Journal of Computer and System Sciences*, 26(2):197–208, 1983.
4. Reuven Bar-Yehuda and Shimon Even. A linear-time approximation algorithm for the weighted vertex cover problem. *Journal of Algorithms*, 2(2):198–203, 1981.
5. Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. A subexponential parameterized algorithm for proper interval completion. *SIAM Journal on Discrete Mathematics*, 29(4):1961–1987, 2015.
6. Ivan Bliznets, Fedor V. Fomin, Marcin Pilipczuk, and Michal Pilipczuk. Subexponential parameterized algorithm for interval completion. *ACM Transactions on Algorithms*, 14(3):35:1–35:62, 2018.
7. Leizhen Cai. Fixed-parameter tractability of graph modification problems for hereditary properties. *Information Processing Letters*, 58(4):171–176, 1996.
8. Leizhen Cai and Chengwei Guo. Contracting few edges to remove forbidden induced subgraphs. In *IPEC*, pages 97–109, 2013.
9. Yixin Cao. Linear recognition of almost interval graphs. In *Proceedings of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1096–1115, Arlington, Virginia, 2016. ACM-SIAM.
10. Yixin Cao. Unit interval editing is fixed-parameter tractable. *Information and Computation*, 253:109–126, 2017.
11. Yixin Cao and Dániel Marx. Interval deletion is fixed-parameter tractable. *ACM Transactions on Algorithms*, 11(3):21:1–21:35, 2015.
12. Yixin Cao and Dániel Marx. Chordal editing is fixed-parameter tractable. *Algorithmica*, 75(1):118–137, 2016.
13. Parinya Chalermsook, Marek Cygan, Guy Kortsarz, Bundit Laekhanukit, Pasin Manurangsi, Danupon Nanongkai, and Luca Trevisan. From Gap-ETH to FPT-inapproximability: Clique, dominating set, and more. In *FOCS*, pages 743–754, 2017. doi:10.1109/FOCS.2017.74.
14. Marek Cygan, Fedor V. Fomin, Łukasz Kowalik, Daniel Lokshtanov, Dániel Marx, Marcin Pilipczuk, Michal Pilipczuk, and Saket Saurabh. *Parameterized Algorithms*. Springer International Publishing, Switzerland, 2015.
15. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer-Verlag Berlin Heidelberg, Germany, 2012.
16. Rod G. Downey and Michael R. Fellows. *Fundamentals of Parameterized complexity*. Springer-Verlag, London, 2013.

17 Pål Grønås Drange, Markus Sortland Dregi, Daniel Lokshtanov, and Blair D. Sullivan. On the threshold of intractability. In *Algorithms - 23rd Annual European Symposium (ESA)*, pages 411–423, Germany, 2015. Springer-Verlag Berlin Heidelberg.

18 Pål Grønås Drange, Fedor V. Fomin, Michal Pilipczuk, and Yngve Villanger. Exploring the subexponential complexity of completion problems. *ACM Transactions on Computation Theory*, 7(4):14:1–14:38, 2015.

19 Pål Grønås Drange and Michal Pilipczuk. A polynomial kernel for trivially perfect editing. *Algorithmica*, 80:3481–3524, 2018.

20 Pavel Dvorák, Andreas Emil Feldmann, Dusan Knop, Tomás Masarík, Tomas Toufar, and Pavel Veselý. Parameterized approximation schemes for steiner trees with small number of steiner vertices. In *STACS*, pages 26:1–26:15, 2018.

21 Eduard Eiben, Danny Hermelin, and M. S. Ramanujan. Lossy kernels for hitting subgraphs. In *MFCS*, pages 67:1–67:14, 2017. doi:10.4230/LIPIcs.MFCS.2017.67.

22 Eduard Eiben, Mithilesh Kumar, Amer E Mouawad, Fahad Panolan, and Sebastian Siebertz. Lossy kernels for connected dominating set on sparse graphs. *SIAM Journal on Discrete Mathematics*, 33(3):1743–1771, 2019.

23 Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Texts in Theoretical Computer Science. An EATCS Series. Springer-Verlag Berlin Heidelberg, Germany, 2006.

24 Fedor V Fomin, Stefan Kratsch, Marcin Pilipczuk, Michał Pilipczuk, and Yngve Villanger. Tight bounds for parameterized complexity of cluster editing with a small number of clusters. *Journal of Computer and System Sciences*, 80(7):1430–1447, 2014.

25 Fedor V Fomin, Daniel Lokshtanov, Saket Saurabh, and Meirav Zehavi. *Kernelization: theory of parameterized preprocessing*. Cambridge University Press, 2019.

26 Fedor V Fomin and Yngve Villanger. Subexponential parameterized algorithm for minimum fill-in. *SIAM Journal on Computing*, 42(6):2197–2216, 2013.

27 Esha Ghosh, Sudeshna Kolay, Mrinal Kumar, Pranabendu Misra, Fahad Panolan, Ashutosh Rai, and M. S. Ramanujan. Faster parameterized algorithms for deletion to split graphs. *Algorithmica*, 71(4):989–1006, 2015.

28 Petr A. Golovach, Pim van 't Hof, and Daniel Paulusma. Obtaining planarity by contracting few edges. *Theoretical Computer Science*, 476:38–46, 2013.

29 Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs*, volume 57. Elsevier, Academic Press, 2004.

30 Sylvain Guillemot and Dániel Marx. A faster FPT algorithm for bipartite contraction. *Information Processing Letters*, 113(22–24):906–912, 2013.

31 Spoorthy Gunda, Pallavi Jain, Daniel Lokshtanov, Saket Saurabh, and Prafullkumar Tale. On the parameterized approximability of contraction to classes of chordal graphs, 2020. arXiv:2006.10364.

32 Chengwei Guo and Leizhen Cai. Obtaining split graphs by edge contraction. *Theoretical Computer Science*, 607:60–67, 2015.

33 Pinar Heggernes, Pim van 't Hof, Benjamin Lévêque, Daniel Lokshtanov, and Christophe Paul. Contracting graphs to paths and trees. *Algorithmica*, 68(1):109–132, 2014.

34 Pinar Heggernes, Pim van 't Hof, Daniel Lokshtanov, and Christophe Paul. Obtaining a bipartite graph by contracting few edges. *SIAM Journal on Discrete Mathematics*, 27(4):2143–2156, 2013.

35 Karthik C. S., Bundit Laekhanukit, and Pasin Manurangsi. On the parameterized complexity of approximating dominating set. In *STOC*, pages 1283–1296. ACM, 2018. doi:10.1145/3188745.3188896.

36 R Krithika, Diptapriyo Majumdar, and Venkatesh Raman. Revisiting connected vertex cover: Fpt algorithms and lossy kernels. *Theory of Computing Systems*, 62(8):1690–1714, 2018.

37 R Krithika, Pranabendu Misra, Ashutosh Rai, and Prafullkumar Tale. Lossy kernels for graph contraction problems. In *FSTTCS*, pages 23:1–23:14, 2016.

38 Daniel Lokshtanov, Neeldhara Misra, and Saket Saurabh. On the hardness of eliminating small induced subgraphs by contracting edges. In *IPEC*, pages 243–254, 2013.

39 Daniel Lokshtanov, Pranabendu Misra, Fahad Panolan, Geevarghese Philip, and Saket Saurabh. A $(2+\epsilon)$-factor approximation algorithm for split vertex deletion. In *47th International Colloquium on Automata, Languages and Programming, ICALP 2020*, volume 168 of *LIPIcs*, page to appear, 2020.

40 Daniel Lokshtanov, Fahad Panolan, M. S. Ramanujan, and Saket Saurabh. Lossy kernelization. In *STOC*, pages 224–237, 2017.

41 Pasin Manurangsi. A note on max k-vertex cover: Faster FPT-AS, smaller approximate kernel and improved approximation. In *SOSA*, 2019.

42 Rolf Niedermeier. *Invitation to fixed-parameter algorithms*. Oxford Lecture Series in Mathematics and Its Applications. Oxford University Press, Oxford, 2006.

43 M. S. Ramanujan. An approximate kernel for connected feedback vertex set. In *27th Annual European Symposium on Algorithms, ESA 2019, September 9-11, 2019, Munich/Garching, Germany*, volume 144 of *LIPIcs*, pages 77:1–77:14, 2019.

44 Sebastian Siebertz. Lossy kernels for connected distance-r domination on nowhere dense graph classes. *arXiv preprint*, 2017. `arXiv:1707.09819`.

45 René van Bevern, Till Fluschnik, and Oxana Yu Tsidulko. On $(1+\epsilon)$-approximate problem kernels for the rural postman problem. *arXiv preprint*, 2018. `arXiv:1812.10131`.

46 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the removal of forbidden graphs by edge-deletion or by edge-contraction. *Discrete Applied Mathematics*, 3(2):151–153, 1981.

47 Toshimasa Watanabe, Tadashi Ae, and Akira Nakamura. On the NP-hardness of edge-deletion and-contraction problems. *Discrete Applied Mathematics*, 6(1):63–78, 1983.

Online Coloring of Short Intervals

Joanna Chybowska-Sokół
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
j.sokol@mini.pw.edu.pl

Grzegorz Gutowski
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
gutowski@tcs.uj.edu.pl

Konstanty Junosza-Szaniawski
Faculty of Mathematics and Information Science, Warsaw University of Technology, Poland
k.szaniawski@mini.pw.edu.pl

Patryk Mikos
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
mikos@tcs.uj.edu.pl

Adam Polak
Institute of Theoretical Computer Science, Faculty of Mathematics and Computer Science, Jagiellonian University, Kraków, Poland
polak@tcs.uj.edu.pl

Abstract

We study the online graph coloring problem restricted to the intersection graphs of intervals with lengths in $[1, \sigma]$. For $\sigma = 1$ it is the class of unit interval graphs, and for $\sigma = \infty$ the class of all interval graphs. Our focus is on intermediary classes. We present a $(1 + \sigma)$-competitive algorithm, which beats the state of the art for $1 < \sigma < 2$, and proves that the problem we study can be strictly easier than online coloring of general interval graphs. On the lower bound side, we prove that no algorithm is better than 5/3-competitive for any $\sigma > 1$, nor better than 7/4-competitive for any $\sigma > 2$, and that no algorithm beats the 5/2 asymptotic competitive ratio for all, arbitrarily large, values of σ. That last result shows that the problem we study can be strictly harder than unit interval coloring. Our main technical contribution is a recursive composition of strategies, which seems essential to prove any lower bound higher than 2.

2012 ACM Subject Classification Theory of computation → Online algorithms

Keywords and phrases Online algorithms, graph coloring, interval graphs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.52

Category APPROX

Related Version A full version of the paper is available at `https://arxiv.org/abs/1802.09503`.

Funding *Joanna Chybowska-Sokół*: Partially supported by the National Science Center of Poland under grant no. 2016/23/N/ST1/03181.
Grzegorz Gutowski: Partially supported by the National Science Center of Poland under grant no. 2016/21/B/ST6/02165.
Patryk Mikos: Partially supported by the National Science Center of Poland under grant no. 2014/14/A/ST6/00138.
Adam Polak: Partially supported by the Polish Ministry of Science and Higher Education program *Diamentowy Grant* under grant no. DI2012 018942.



© Joanna Chybowska-Sokół, Grzegorz Gutowski, Konstanty Junosza-Szaniawski, Patryk Mikos, and Adam Polak;
licensed under Creative Commons License CC-BY



1 Introduction

In the *online graph coloring* problem the input graph is presented to the algorithm vertex by vertex, along with all the edges adjacent to the already presented vertices. Each vertex must be assigned a color, different than any of its neighbors, immediately and irrevocably at the moment it is presented, without any knowledge of the remaining part of the graph. The objective is to minimize the number of used colors. The problem and its variants attract much attention, both for theoretical properties and practical applications in network multiplexing, resource allocation, and job scheduling.

The standard performance measure, used to analyze online algorithms, is the *competitive ratio*, i.e., the worst-case guarantee on the ratio of the solution given by an online algorithm to the optimal offline solution (see Section 1.1 for a formal definition).

In the general case, of online coloring of arbitrary graphs, there is no hope for any algorithm with a constant competitive ratio. The best known algorithm [7] uses $O(\chi \cdot n/\log n)$ colors for n-vertex χ-colorable graphs, i.e. it is $O(n/\log n)$-competitive, and there is a lower bound [8] showing that no online graph coloring algorithm can be $o\big(n/\log^2 n\big)$-competitive. It is thus common to study the problem restricted to particular graph classes.

Having in mind the applications in scheduling, one of the important special cases is the class of *interval graphs*, i.e. intersection graphs of intervals on the real line. The classic result is by Kierstead and Trotter [11], who designed a 3-competitive algorithm and proved a matching lower bound. However, in the special case of *unit interval graphs*, i.e. intersection graphs of intervals of a fixed length, already the simple greedy FirstFit algorithm is 2-competitive [4].

Coloring unit interval graphs can model only a restricted scheduling setting, with all jobs having the same processing time. On the other hand, allowing arbitrary processing times, modeled by (general) interval graphs, might be a too permissive setting, precluding efficient online algorithms. Thus, we ask what happens in between the interval and unit interval graph classes. In particular, we are interested in the optimal competitive ratio of online coloring algorithms for intersection graphs of intervals of length restricted to a fixed range. Formally, let us introduce the σ-interval coloring problem.

▶ **Definition 1.** *For $\sigma \geqslant 1$, the* σ-interval coloring *problem asks: Given a sequence of closed intervals $[l_1, r_1], [l_2, r_2], \ldots, [l_n, r_n]$, such that $1 \leqslant (r_i - l_i) \leqslant \sigma$ for every $i \in [n]$, find a sequence of* colors*, $c_1, c_2, \ldots, c_n$, such that*

$$\forall_{i \neq j} \ \big([l_i, r_i] \cap [l_j, r_j] \neq \emptyset\big) \implies (c_i \neq c_j),$$

minimizing the number of distinct colors $\big|\{c_1, c_2, \ldots, c_n\}\big|$.

We study the problem in the online setting, i.e., intervals are presented one by one, in an arbitrary order, and each interval has to be colored immediately and irrevocably after it is presented.

Note that we choose to include the interval representation in the input, instead of presenting the mere graph. It seems a plausible modeling choice given the scheduling applications. Moreover, it lets algorithms exploit geometric properties of the input, and not only structural graph properties. Naturally, any lower bound obtained for this variant of the problem transfers to the harder variant with no interval representation in the input.

Among others, we look to answer the following two apparent questions.

▶ **Question 1.** *Is σ-interval coloring strictly easier than interval coloring?*

▶ **Question 2.** *Is σ-interval coloring strictly harder than unit interval coloring?*

Naturally, we ask these questions in the online setting, where easier (harder) mean smaller (greater) best possible competitive ratio.

1.1 Our Results

Before we state our results, let us give a formal definition of the competitive ratio. In this paper we focus on the *asymptotic* competitive ratio.

▶ **Definition 2.** *Let A be an online graph coloring algorithm, and let $A(\chi)$ denote the maximum number of colors A uses to color any graph which can be colored offline using χ-colors (i.e. its chromatic number is at most χ). We say that A has the asymptotic competitive ratio α (or that A is α-competitive, for short), if* $\limsup_{\chi\to\infty} \frac{A(\chi)}{\chi} \leqslant \alpha$.

Another popular performance measure for online algorithms is the *absolute* competitive ratio, which requires that $\frac{A(\chi)}{\chi} \leqslant \alpha$ holds for *all* χ (and not only in the limit). The choice of the asymptotic, instead of absolute, competitive ratio for our analysis makes things easier for the algorithm and harder for the lower bounds. In our algorithm, sadly, we do not know how to get rid of a constant additive overhead, which vanishes only with growing χ. This is in contrast to the FirstFit and Kierstead-Trotter algorithms, whose claimed competitive ratios are not only asymptotic but also absolute. The good side is, our lower bounds for the asymptotic competitive ratio imply the identical lower bounds for the absolute competitive ratio.

Algorithm. Our positive result is the existence of a $(1+\sigma)$-competitive algorithm.

▶ **Theorem 3.** *For every $\sigma \in \mathbb{Q}$, $\sigma \geqslant 1$, there is an algorithm for online σ-interval coloring with $1+\sigma$ asymptotic competitive ratio.*

Note that for $\sigma' > \sigma$ every σ'-interval coloring algorithm is also a correct σ-interval coloring algorithm, with the same upper bound on its competitive ratio. Therefore, for $\sigma \in \mathbb{R} \setminus \mathbb{Q}$ Theorem 3 yields an online σ-interval coloring algorithm with a competitive ratio arbitrarily close to $1+\sigma$. This distinction between rational and irrational values of σ becomes somewhat less peculiar in the light of the results of Fishburn and Graham [5], who proved, among other things, that the classes $C(\sigma)$'s of graphs with interval representation with lengths in $[1,\sigma]$ are *right-continuous*, i.e. $C(\sigma) = \bigcap_{\tau>\sigma} C(\tau)$, exactly at irrational σ's.

Until now, the state-of-the art was the 2-competitive FirstFit algorithm [4] for $\sigma = 1$, and the 3-competitive Kierstead-Trotter algorithm [11] for $\sigma > 1$. Our algorithm matches the performance of FirstFit for $\sigma = 1$, and beats the Kierstead-Trotter algorithm for $\sigma < 2$.

The algorithm is inspired by the recent result for online coloring of unit disk intersection graphs [9]. We cover the real line with overlapping blocks, grouped into a constant number of classes, with each class constituting a partition of the real line. We assign to each class a private set of available colors. When an interval is presented, the algorithm chooses, in a round-robin fashion, a block containing the interval's left end, and greedily picks a color from the block's class.

While not being overly complicated, the algorithm already answers positively our Question 1. Indeed, since there cannot exist a less-than-3-competitive algorithm for (general) interval coloring [11], σ-interval coloring is strictly easier, for $\sigma < 2$.

Lower Bounds. Our negative results include a series of constructions with the following consequences.

▶ **Theorem 4.** *For every $\sigma > 1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.*

▶ **Theorem 5.** *For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $7/4$.*

▶ **Theorem 6.** *For every $\varepsilon > 0$ there is $\sigma \geqslant 1$ such that there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio $5/2 - \varepsilon$.*

The following, more illustrative, statement is a direct corollary of Theorem 6.

▶ **Corollary 7.** *There is no online algorithm that works for all $\sigma \geqslant 1$ and uses at most $2.499 \cdot \omega + f(\sigma)$ colors for ω-colorable graphs (for any function f).*

Theorem 6 gives a positive answer to our Question 2. Indeed, while FirstFit is 2-competitive for unit interval coloring, there is no 2-competitive algorithm for σ-interval coloring (for large enough σ), therefore the latter problem is strictly harder. Working out the exact number from our proof, this starts to be the case for $\sigma > 2^{78}$, however we did not attempt to optimize the constant.

Our proofs of Theorems 4 and 5 can be considered as generalizations of the $3/2$ lower bound for online coloring of unit interval graphs by Epstein and Levy [4]. In particular, we heavily use their *separation strategy*, which also appears in [1].

Our main technical contribution is a recursive composition of strategies, which seems essential to prove any lower bound higher than 2. Our $5/2$ lower bound (Theorem 6) borrows also from the work of Kierstead and Trotter [11]. However, in order to control the length of intervals independently of the number of colors, we cannot simply use the pigeonhole principle, as they did. Instead, we develop Lemmas 19 and 20, which let us overcome this issue, at a cost of a worse bound for the competitive ratio, i.e. $5/2$ instead of 3.

1.2 Related Work

Interval graphs have been intensively studied since the sixties [2, 12], and, in particular, they are known to be *perfect*, i.e. the chromatic number χ of an interval graph always equals the size of the largest clique ω (see, e.g., [6]). To construct an optimal coloring offline it is sufficient to color the graph greedily in a nondecreasing order of the left ends of the intervals.

For the most basic approach for online coloring, that is the FirstFit algorithm, the competitive ratio for interval graphs is still not known exactly. After a series of papers, the most recent results state that FirstFit is at least 5- and at most 8-competitive [10, 13]. Kierstead and Trotter [11] designed a more involved online coloring algorithm, which uses at most $3\omega - 2$ colors for ω-colorable interval graphs, and proved that there exists a strategy that forces any online coloring algorithm to use exactly that number of colors. The same lower and upper bounds were obtained independently by Chrobak and Ślusarek [3, 14]. For intersection graphs of intervals of unit length any online coloring algorithm uses at least $\frac{3}{2}\omega$ colors, and FirstFit uses at most $2\omega - 1$ colors [4].

It seems a natural question to ask if it is possible to improve the bound of $3\omega - 2$ by assuming that interval lengths belong to a fixed range. The study of interval graphs with bounded length representations was initiated by Fishburn and Graham [5]. However, it focused mainly on the combinatorial structure, and not its algorithmic applications.

Kierstead and Trotter [11] give, for every $\omega \in \mathbb{N}_+$, a strategy for Presenter to construct an ω-colorable set of intervals while forcing Algorithm to use at least $3\omega - 2$ colors. However, the lengths of presented intervals increase with the increasing ω. For this reason, with the

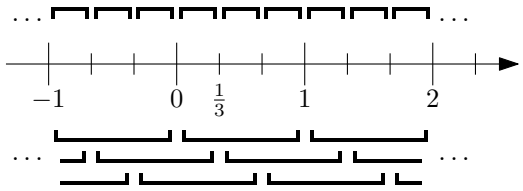


Figure 1 Small blocks (above), and large blocks (below), for $b = 3$.

interval length restricted to $[1, \sigma]$, their lower bound is only for the absolute competitive ratio and does not exclude, say, an algorithm that always uses at most $2\omega + \sigma^{10}$ colors. On the contrary, in Theorem 6 we rule out the existence of such an algorithm.

2 Algorithm

▶ **Theorem 3.** *For every $\sigma \in \mathbb{Q}$, $\sigma \geqslant 1$, there is an algorithm for online σ-interval coloring with $1 + \sigma$ asymptotic competitive ratio.*

Proof. Let us present an algorithm which, in principle, works for any real σ, however only for a rational σ it achieves the declared competitive ratio. The algorithm has a positive integer parameter b. Increasing the parameter brings the asymptotic competitive ratio closer to $1 + \sigma$ at the cost of increasing the additive constant. More precisely, given an ω-colorable set of intervals our algorithm colors it using at most $\lceil b \cdot (1 + \sigma) \rceil \cdot \left(\frac{\omega}{b} + b - 1\right)$ colors, and thus its competitive ratio is $\frac{\lceil b \cdot (1+\sigma) \rceil}{b} + O(1/\omega)$. For a rational σ, in order to obtain exactly the declared $1 + \sigma$ asymptotic competitive ratio it is sufficient to set b to the smallest possible denominator of a simple fraction representation of σ. Let $\varphi = \lceil b \cdot (1 + \sigma) \rceil$. The algorithm will use colors from the set $\{0, 1, \ldots, \varphi - 1\} \times \mathbb{N}$.

Now, let us consider the partition of the real line into *small blocks*. For $i \in \mathbb{Z}$, the i-th small block occupies the interval $[i \cdot \frac{1}{b}, (i + 1) \cdot \frac{1}{b})$. Moreover, we define *large blocks*. The i-th large block occupies the interval $[i \cdot \frac{1}{b}, i \cdot \frac{1}{b} + 1)$. See Figure 1.

Let us point out certain properties of the blocks, which will be useful in the further analysis. Each large block is the union of b consecutive small blocks, and each small block is a subset of b consecutive large blocks. Further, the length of a large block is 1, and for any two intervals of length in $[1, \sigma]$ that both have the left endpoint in the same large block, the two intervals intersect. Thus, the intervals whose left endpoints belong to a fixed large block form a clique. Finally, if the indices of two large blocks differ by at least φ, then any two intervals – one with the left endpoint in one block, the other with the left endpoint in the other – do not intersect.

With each small block the algorithm associates a *small counter*, and with each large block the algorithm associates a *large counter*. Let S_i denote the small counter of the i-th small block, and L_j denote the large counter of the j-th large block. Initially, all the small and large counters are set to zero.

The small counter S_i tracks how many intervals with left endpoint in the i-th small block appeared so far. Based on the small counter the algorithm assigns, in a round-robin fashion, the processed interval to one of the b large blocks containing its left endpoint. The large counter L_j tracks how many intervals were assigned to the j-th large block so far. The color of the processed interval will depend on the large counter.

To assign a color to an interval, the algorithm proceeds as follows:

1. Let i be the index of the small block containing the left endpoint of the interval.
2. Let j be the index of the large block containing the left endpoint of the interval such that $j \equiv S_i \pmod{b}$. Note that there is exactly one such j.

3. Assign to the interval the color $(j \bmod \varphi, L_j)$.
4. Increase the small counter S_i by one.
5. Increase the large counter L_j by one.

First let us argue that the algorithm outputs a proper coloring. Consider any two intervals which were assigned the same color. Let j_1 and j_2 denote the indices of the large blocks selected for these intervals by the algorithm. Since the colors of the two intervals have the same first coordinates, we have that $j_1 \equiv j_2 \pmod{\varphi}$. However, since the second coordinates, which are determined by large counters, are also the same, j_1 and j_2 must be different, and thus they differ by at least φ. Hence the left endpoints of the large blocks j_1 and j_2 are at least $1 + \sigma$ apart, and the two considered intervals do not intersect, thus the coloring is proper.

It remains to bound the number of colors in terms of the clique number ω. Let j be the index of the maximum large counter L_j at the end of the algorithm. Clearly, the algorithm used at most $\varphi \cdot L_j$ colors in total. Let C denote the set of intervals with the left endpoints in the j-th large block and colored with a color in $\{j \bmod \varphi\} \times \mathbb{N}$. Observe that $|C| = L_j$. Let x_k denote the number of intervals in C which have the left endpoint in the k-th small block. Recall that the j-th large block is the union of b small blocks – indexed $j, j+1, \ldots, j+b-1$ – and thus $L_j = x_j + x_{j+1} + \cdots + x_{j+b-1}$. Because of the the formula $j \equiv S_i \pmod{b}$ in the step (2) of the algorithm, the large counter L_j is incremented by one out of every b intervals with the left endpoints in any given small block. Hence either $x_k = \lfloor \frac{S_k}{b} \rfloor$ or $x_k = \lfloor \frac{S_k}{b} \rfloor + 1$. In particular

$$S_k \geqslant b \cdot (x_k - 1) + 1.$$

Let D denote the set of all intervals with the left endpoints in the j-th large block. We can bound the number of intervals in D

$$|D| \;=\; \sum_{k=j}^{j+b-1} S_k \;\geqslant\; \sum_{k=j}^{j+b-1} \left(b \cdot (x_k - 1) + 1\right) \;=\; b \cdot (L_j - b) + b.$$

Recall that D is a clique and thus the clique number ω of the input graph is at least the size of D. Therefore $L_j \leqslant \frac{\omega + b \cdot (b-1)}{b}$, and the algorithm used at most

$$\lceil b \cdot (1 + \sigma) \rceil \cdot \left(\frac{\omega}{b} + b - 1 \right)$$

colors. ◀

3 Lower Bounds

3.1 Technical Overview

Algorithm-vs-Presenter game. In order to prove lower bounds for online problems, it is often convenient to look at the problem as a combinatorial game between two players, Algorithm and Presenter. In our case, in each round Presenter reveals an interval, and Algorithm immediately and irrevocably assigns a color to it. Algorithm tries to minimize the number of different colors it assigns. Contrarily, the Presenter's goal is to force Algorithm to use as many colors as possible, while guaranteeing that the introduced set of intervals is colorable with a smaller number of colors, and contains only short intervals. A strategy for Presenter implies a lower bound on the competitive ratio of any algorithm solving the problem.

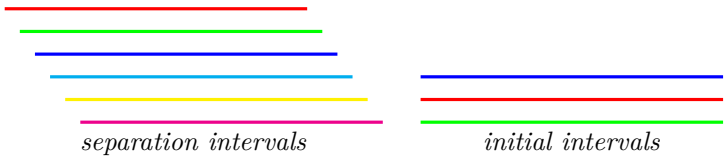


Figure 2 Separation strategy.

Separation Strategy. Epstein and Levy [4] prove their $\frac{3}{2}$ lower bound for online coloring of unit intervals by giving the following strategy for Presenter. The strategy has three phases.

1. Presenter introduces a clique of $\frac{\omega}{2}$ equal (i.e. located at the same place) *initial intervals*. Algorithm has to use $\frac{\omega}{2}$ different colors. Let $\mathcal{X}$ denote the set of these colors.
2. Just to the left of the initial intervals, Presenter introduces a clique of ω *separation intervals*. These intervals are not equal, they all have slightly different endpoints, and in total they occupy a place of length $1+\varepsilon$. None of them intersect the initial intervals. The specific way the separation intervals are presented ensures that all separation intervals with colors in $\mathcal{X}$ have their left (resp. right) endpoints to the left of all left (resp. right) endpoints of separation intervals with colors not in $\mathcal{X}$ (see Figure 2). In particular, each of $\frac{\omega}{2}$ right-most separation intervals has a color not in $\mathcal{X}$.
3. Presenter introduces a clique of equal $\frac{\omega}{2}$ *final intervals* that intersect all the initial intervals, and $\frac{\omega}{2}$ right-most separation intervals.

Algorithm has to use at least $\frac{3}{2}\omega$ different colors, but the largest clique size (i.e. the offline optimum) is only ω.

The second phase of the above strategy is an example of the *separation strategy*. The formal details are included in the proof of Lemma 14.

Recursive Composition of Strategies. Let us generalize the above strategy. Observe, that instead of presenting a clique in the first phase, Presenter can use an arbitrary strategy, presenting a set of intervals with the clique number equal to $\frac{\omega}{2}$ but possibly enforcing more than $\frac{\omega}{2}$ colors. Moreover, Presenter might be able to achieve a better trade-off by allowing a clique of size $\beta\omega$ (for some $0 < \beta < 1$) in the first phase, and introducing a clique of size $(1-\beta)\omega$ in the third phase. Actually, it turns out that if the strategy used in the first phase enforces a competitive ratio α (e.g. presenting a clique has $\alpha = 1$), then it is optimal to set $\beta = \frac{1}{1+\alpha}$.

Assume we have a strategy that enforces a competitive ratio α, uses intervals of length at most σ, and needs a place of length M on the real axis (e.g. presenting a clique gives $\alpha = \sigma = M = 1$). Consider the following strategy.

1. Presenter plays the assumed strategy to obtain a set of initial intervals with the clique number $\frac{\omega}{1+\alpha}$. Algorithm has to use $\alpha\frac{\omega}{1+\alpha}$ different colors, denoted by $\mathcal{X}$.
2. Presenter introduces a clique of ω separation intervals (of unit length), and at least the right-most $\omega - |\mathcal{X}| \geqslant \frac{\omega}{1+\alpha}$ of them get new colors, not in $\mathcal{X}$.
3. Presenter introduces a clique of $\alpha\frac{\omega}{1+\alpha}$ final intervals of length $M+\varepsilon$ that intersect all the initial intervals, and the $\frac{\omega}{1+\alpha}$ right-most separation intervals.

In total, the intervals can be colored with ω colors, but the algorithm uses at least

$$\alpha\frac{\omega}{1+\alpha} + \frac{\omega}{1+\alpha} + \alpha\frac{\omega}{1+\alpha} = \frac{(2\alpha+1)\omega}{1+\alpha}$$

colors, i.e. it is at most $(2 - \frac{1}{\alpha+1})$-competitive

Let S_0 be a trivial strategy that presents a clique of equal unit intervals. Now, let S_{i+1} be a strategy obtained as above by playing S_i in the first phase. It enforces a competitive ratio α_{i+1}, uses intervals of length at most σ_{i+1}, and needs a place of length M_{i+1}, where $\alpha_0 = \sigma_0 = M_0 = 1$, and each $\alpha_{i+1} = 2 - \frac{1}{\alpha_i+1}$, $\sigma_{i+1} = M_i + \varepsilon$, $M_{i+1} = M_i + 1 + \varepsilon$. By solving the recurrence equations we get the following corollary.

▶ **Corollary 8.** *For every $n \in \mathbb{N}_+$ and every $\varepsilon > 0$, there is no online algorithm for $(n+\varepsilon)$-interval coloring with the asymptotic competitive ratio less than $\frac{F_{2n+1}}{F_{2n}}$, where F_n is the n-th Fibonacci number ($F_0 = F_1 = 1$, $F_{n+2} = F_{n+1} + F_n$).*

Note that this method cannot give a lower bound higher than $\lim_{n\to\infty} \frac{F_{2n+1}}{F_{2n}} = \frac{1+\sqrt{5}}{2} \approx 1.61803$. However, we can get arbitrarily close to this bound. That is, for every $\varepsilon > 0$ there is a σ and ω_0 such that for each $\omega \geqslant \omega_0$ there is a strategy for Presenter to present intervals of length at most σ and force Algorithm to use $\left(\frac{1+\sqrt{5}}{2} - \varepsilon\right) \cdot \omega$ colors on an ω-colorable set of intervals.

▶ **Observation 9.** *There is no online algorithm that works for all $\sigma \geqslant 1$ and uses at most $1.618 \cdot \omega + f(\sigma)$ colors for ω-colorable graphs (for any function f).*

Overview of the 5/2 Lower Bound. Our $5/2$ lower bound also works by recursively combining strategies, however the three phases are substantially different than above. In particular, in the first phase, instead of playing the simpler strategy only once, Presenter plays many instances of it, independently, side by side. Intuitively, either Algorithm uses mostly different colors for different instances, and it already uses too many colors, or there must be many colors shared by many instances.

So far the argument resembles the lower bound for interval graphs by Kierstead and Trotter [11]. They use pigeonhole principle to argue that if the total number of colors is bounded, then after playing exponentially many instances, there must be four instances that use exactly the same subset of colors. Thus, the number of instances – and consequently the length of intervals in the subsequent phases – required in their approach grows with the number of colors, which makes it infeasible in our setting.

To overcome the above difficulty we show (see Lemmas 19 and 20) that, if the total number of colors is small, there must be four groups of consecutive instances, which do not necessarily use exactly the same colors, but at least have large intersection.

3.2 Preliminaries

To properly capture asymptotic properties of the introduced strategies we give the following formal definitions.

▶ **Definition 10.** *For $\omega, C \in \mathbb{N}_+$ and $\sigma, M \in \mathbb{R}_+$, an* $\langle \omega, C, \sigma, M\rangle$-strategy *is a strategy for Presenter that forces Algorithm to use at least C colors subject to the following constraints:*

1. *the set of introduced intervals is ω-colorable,*
2. *every introduced interval has length at least 1 and at most σ,*
3. *every introduced interval is contained in the interval $[0, M]$.*

We are interested in providing strategies that achieve the biggest possible ratio $\frac{C}{\omega}$ for large ω. This motivates the following definition.

▶ **Definition 11.** *An* $\langle \alpha, \sigma, M\rangle$-schema *is a set of $\langle \omega, C_\omega, \sigma, M\rangle$-strategies for all $\omega \in \mathbb{N}_+$ such that $C_\omega = \alpha\omega - o(\omega)$.*

The $o(\omega)$ term in the above definition accounts for the fact that sometimes in a proof we would like to introduce, say, $\frac{\omega}{2}$-clique. Then, for odd ω's a rounding is required, which results in small inaccuracies we need to control.

▶ Remark 12. Note that the existence of an $\langle \alpha, \sigma, M \rangle$-*schema* implies a lower bound of α for the asymptotic competitive ratio of any online algorithm solving the σ-interval coloring problem.

To put the above definitions in context, note that Kierstead and Trotter [11] give, for all $\omega \in \mathbb{N}_+$, an $\langle \omega, 3\omega - 2, f(\omega), f(\omega) \rangle$-strategy. However, their family of strategies does not yield an $\langle \alpha, \sigma, M \rangle$-schema, because the length of the presented intervals grows with ω.

▶ **Example 13** ($\langle 1, 1, 1 \rangle$-schema)**.** For any $\omega \in \mathbb{N}_+$, a strategy that introduces the interval $[0, 1]$ in every round $1, \ldots, \omega$ is an $\langle \omega, \omega, 1, 1 \rangle$-strategy. The set of these strategies is a $\langle 1, 1, 1 \rangle$-schema.

In this section we show a series of constructions that use an existing schema to create another schema with different parameters. The $\langle 1, 1, 1 \rangle$-schema given above is the initial step for those constructions.

Let S be an $\langle \omega, C, \sigma, M \rangle$-strategy. We will say that *Presenter uses strategy S in the interval* $[x, x + M]$ to denote that Presenter plays according to S, presenting intervals shifted by x, until Algorithm uses C colors.

3.3 Warm-up

Our first construction is a natural generalization of the strategy for unit intervals given by Epstein and Levy [4], already described in Section 3.1 in an informal way. The construction is surpassed by more involved strategies coming later, but it serves as a gentle introduction to our formal framework.

▶ **Lemma 14.** *If there is an $\langle \alpha, \sigma, M \rangle$-schema, then there is a $\left\langle 2 - \frac{1}{\alpha+1}, M + \varepsilon, M + 1 + \varepsilon \right\rangle$-schema for every $\varepsilon > 0$.*

Proof. To prove the lemma we need to provide an $\left\langle \omega, (2 - \frac{1}{\alpha+1})\omega - o(\omega), M + \varepsilon, M + 1 + \varepsilon \right\rangle$-strategy for every $\omega \in \mathbb{N}_+$. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+1} \right\rfloor$. The $\langle \alpha, \sigma, M \rangle$-schema contains an $\langle \omega', \alpha\omega' - \delta, \sigma, M \rangle$-strategy S for some $\delta = o(\omega')$. The strategy for Presenter consists of three phases (see Figure 3). In the first phase, called the *initial phase*, Presenter uses strategy S inside the interval $[1 + \varepsilon, M + 1 + \varepsilon]$. Let $C = \alpha\omega' - \delta$ and let $\mathcal{X}$ denote the set of C colors used by Algorithm in the initial phase.

The second phase, borrowed from [4, 1], is called the *separation phase*. In this phase, Presenter plays the following separation strategy for ω rounds. Let $l_1 = 0$ and $r_1 = \frac{\varepsilon}{2}$. In the i-th round of the separation phase Presenter introduces the interval $[\frac{l_i + r_i}{2}, \frac{l_i + r_i}{2} + 1]$. If Algorithm colors the interval with one of the colors in $\mathcal{X}$, let $l_{i+1} = \frac{l_i + r_i}{2}$ and $r_{i+1} = r_i$, which means that the next interval will be shifted slightly to the right. Otherwise, let $l_{i+1} = l_i$ and $r_{i+1} = \frac{l_i + r_i}{2}$, which means that the next interval will be shifted slightly to the left.

The above procedure guarantees the following invariant. At the beginning of round i all the previously introduced intervals with a color in $\mathcal{X}$ have their left endpoints to the left of l_i, and, conversely, all the previously introduced intervals with a color not in $\mathcal{X}$ have their left endpoints to the right of r_i. Moreover, all the intervals yet to be introduced will have their left endpoints strictly between l_i and r_i.

Observe that all intervals introduced in the separation phase have length 1 and $\forall_i \frac{l_i + r_i}{2} < \frac{\varepsilon}{2}$. Thus, every interval introduced in the separation phase is contained in $\left[0, 1 + \frac{\varepsilon}{2}\right]$ and any two of those intervals intersect. Furthermore, the above invariant guarantees that for any two

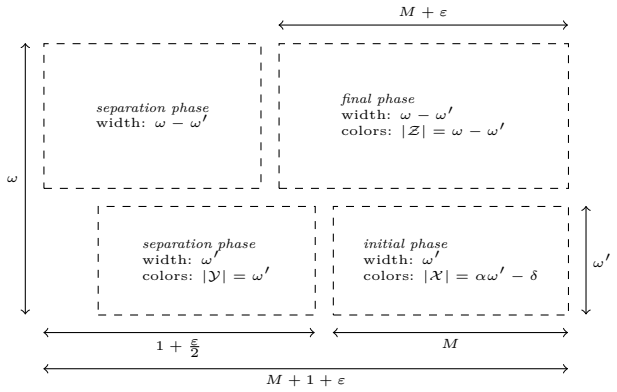


Figure 3 Strategy construction in Lemma 14.

intervals x, y introduced in the separation phase, x colored with a color in $\mathcal{X}$, and y colored with a color not in $\mathcal{X}$, we have that the left endpoint of x is to the left of the left endpoint of y. Let Y be the set of $\omega' \leqslant \omega - |\mathcal{X}|$ right-most intervals introduced in the separation phase, and let $\mathcal{Y}$ be the set of colors used by Algorithm on the intervals in Y. Note that $\mathcal{X}$ and $\mathcal{Y}$ are disjoint.

For the last phase, called the *final phase*, let r be the left-most right endpoint of an interval in Y. In the final phase Presenter introduces $\omega - \omega'$ times the same interval $[r, M+1+\varepsilon]$. This interval intersects all intervals introduced in the initial phase, all intervals in Y, and no other interval introduced in the separation phase. Thus, Algorithm must use $\omega - \omega'$ colors in the final phase that are different from the colors in both $\mathcal{X}$ and $\mathcal{Y}$. Let $\mathcal{Z}$ denote the set of colors used by Algorithm in the final phase.

The presented set of intervals is clearly ω-colorable and Algorithm used at least $|\mathcal{X}| + |\mathcal{Y}| + |\mathcal{Z}| = \alpha\omega' - \delta + \omega' + \omega - \omega' = \left(2 - \frac{1}{\alpha+1}\right)\omega - o(\omega)$ many colors. The longest interval presented has length $M+\varepsilon$, and all intervals are contained in $[0, M+1+\varepsilon]$. Thus, we have constructed a $\left\langle 2 - \frac{1}{\alpha+1}, M+\varepsilon, M+1+\varepsilon \right\rangle$-schema. ◀

3.4 The 5/3 Lower Bound

▶ **Lemma 15.** *If there is an $\langle \alpha, \sigma, M\rangle$-schema, then there is a $\left\langle 2 - \frac{1}{\alpha+2}, M+\varepsilon, M+2+\varepsilon \right\rangle$-schema for every $\varepsilon > 0$.*

Proof. The proof of this lemma is very similar to the proof of Lemma 14, but now we have two separation phases instead of just one, see Figure 4. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+2} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M\rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses S inside interval $\left[1+\frac{\varepsilon}{2}, M+1+\frac{\varepsilon}{2}\right]$, and forces Algorithm to use $C = \alpha\omega' - \delta$ colors. Let $\mathcal{X}$ denote the set of those colors.

In the separation phase, Presenter plays the separation strategy two times. First, Presenter plays the separation strategy for ω rounds in the region $\left[0, 1+\frac{\varepsilon}{4}\right]$ pushing to the right colors not in $\mathcal{X}$. Let Y_1 be the set of ω' right-most intervals from this first separation. Let $\mathcal{Y}_1$ denote the set of colors used by Algorithm to color Y_1. Then, Presenter plays the separation strategy for ω rounds in the region $\left[M+1+\frac{3\varepsilon}{4}, M+2+\varepsilon\right]$ pushing to the left colors not in $\mathcal{X} \cup \mathcal{Y}_1$. Let Y_2 be the set of ω' left-most intervals from this second separation. Let $\mathcal{Y}_2$ denote the set of colors used by Algorithm to color Y_2.

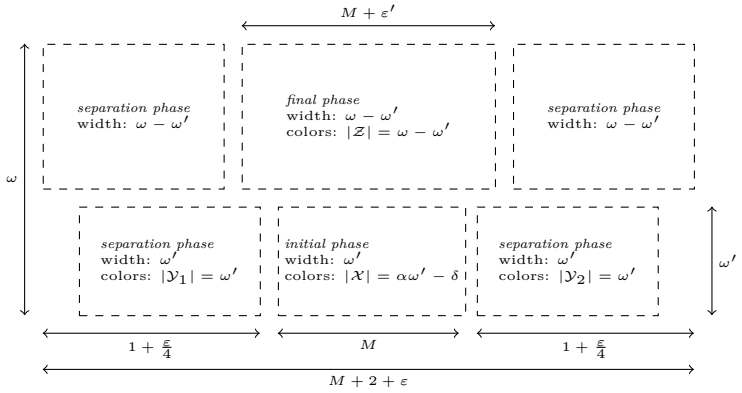


Figure 4 Strategy construction in Lemma 15.

Let r be the left-most right endpoint of an interval in Y_1. Let l be the right-most left endpoint of an interval in Y_2. In the final phase Presenter introduces $\omega - \omega'$ times the same interval $[r, l]$.

The presented set of intervals is clearly ω-colorable and Algorithm used at least $|\mathcal{X}| + |\mathcal{Y}_1| + |\mathcal{Y}_2| + |\mathcal{Z}| = \alpha\omega' - \delta + \omega' + \omega' + \omega - \omega' = \left(2 - \frac{1}{\alpha+2}\right)\omega - o(\omega)$ many colors. The longest interval presented has length at most $M + \varepsilon$, and all intervals are contained in $[0, M + 2 + \varepsilon]$. Thus, we have constructed a $\left\langle 2 - \frac{1}{\alpha+2}, M + \varepsilon, M + 2 + \varepsilon \right\rangle$-schema. ◀

▶ **Corollary 16.** *There is an $\langle \alpha_n, 2n - 1 + \varepsilon, 2n + 1 + \varepsilon \rangle$-schema, for every $n \in \mathbb{N}_+$ and every $\varepsilon > 0$, where*

$$\alpha_n = \frac{\left(\sqrt{3} - 3\right)\left(\sqrt{3} - 2\right)^n + \left(\sqrt{3} + 3\right)\left(-\sqrt{3} - 2\right)^n}{\left(\sqrt{3} - 1\right)\left(\sqrt{3} - 2\right)^n + \left(\sqrt{3} + 1\right)\left(-\sqrt{3} - 2\right)^n}.$$

Proof. Starting with a $\langle 1, 1, 1 \rangle$-schema and repeatedly applying Lemma 15 one can generate[1] a family of $\langle \alpha_n, \sigma_n + \varepsilon_n, M_n + \varepsilon_n \rangle$-schemas, such that $\alpha_{n+1} = 2 - \frac{1}{\alpha_n + 2}$, $\sigma_{n+1} = M_n$, $M_{n+1} = M_n + 2$, and $\alpha_0 = \sigma_0 = M_0 = 1$. Solving the recurrence equations we get $\alpha_n = \frac{F_{2n+1}}{F_{2n}}$, $\sigma_n = 2n - 1$, and $M_n = 2n + 1$. ◀

Note that, similarly to Observation 9, one could already use Corollary 16 to get a lower bound arbitrarily close to $\lim_{n\to\infty} \alpha_n = \sqrt{3} \approx 1.73205$ for the asymptotic competitive ratio of any online algorithm that work for all $\sigma \geqslant 1$. Nonetheless in Section 3.6 we prove a stronger $5/2$ lower bound.

▶ **Theorem 4.** *For every $\sigma > 1$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $5/3$.*

Proof. Assume for contradiction that for some $\sigma > 1$ there exists an online algorithm for σ-interval coloring with the asymptotic competitive ratio $\frac{5}{3} - \varepsilon$, for some $\varepsilon > 0$. By the definition of the asymptotic competitive ratio, there is an ω_A such that for every $\omega \geqslant \omega_A$ the algorithm colors every ω-colorable set of intervals using at most $\left(\frac{5}{3} - \varepsilon + \frac{\varepsilon}{3}\right)\omega = \left(\frac{5}{3} - \frac{2\varepsilon}{3}\right)\omega$ colors.

[1] Knowing the desired target values of n and ϵ, one needs to properly adjust the ϵ value for each application of Lemma 15, e.g., it is sufficient to set it to ϵ/n.

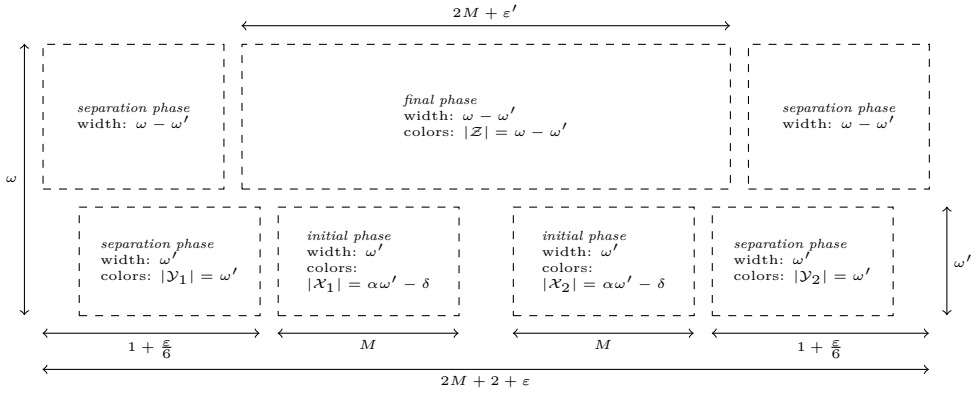


Figure 5 Lemma 17, Case 1: $|\mathcal{C}_2 \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$.

Observe that, for $n = 1$, Corollary 16 gives a $\left\langle \frac{5}{3}, 1 + (\sigma - 1), 3 + (\sigma - 1) \right\rangle$-schema. By the definition of schema, there is an ω_P such that for every $\omega \geqslant \omega_P$ there is a strategy for Presenter to present an ω-colorable set of intervals, of length in $[1, \sigma]$, and force Algorithm to use $\left(\frac{5}{3} - \frac{\varepsilon}{3}\right)\omega$ colors. For $\omega = \max(\omega_A, \omega_P)$ we reach a contradiction. ◀

3.5 The 7/4 Lower Bound

▶ **Lemma 17.** *If there is an $\langle \alpha, \sigma, M \rangle$-schema, then there is a $\left\langle \frac{2\alpha+1}{2\alpha+2}, 2M + \varepsilon, 2M + 2 + \varepsilon \right\rangle$-schema for every $\varepsilon > 0$.*

Proof. The proof of this lemma is a bit more complicated than the previous ones, as we now have two initial phases, two separation phases and a strategy branching, see Figure 5 and Figure 6. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{\alpha+1} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M \rangle$-strategy for some $\delta = o(\omega')$.

In the initial phase, Presenter uses strategy S twice: (1) inside interval $\left[1 + \frac{\varepsilon}{3}, M + 1 + \frac{\varepsilon}{3}\right]$, and (2) inside interval $\left[M + 1 + \frac{2\varepsilon}{3}, 2M + 1 + \frac{2\varepsilon}{3}\right]$. Algorithm uses $C = \alpha\omega' - \delta$ colors in each of these intervals. We get a set of colors $\mathcal{X}_1$ used by Algorithm in the first interval, and a set of colors $\mathcal{X}_2$ used by Algorithm in the second interval. Note that $\mathcal{X}_1 \cap \mathcal{X}_2$ might be non-empty.

In the separation phase, Presenter plays the separation strategy two times. First, Presenter plays the separation strategy for ω rounds in the region $\left[0, 1 + \frac{\varepsilon}{6}\right]$ pushing to the right colors not in $\mathcal{X}_1$. Let Y_1 be the set of ω' right-most intervals from the first separation phase. Let $\mathcal{Y}_1$ denote the set of colors used by Algorithm to color Y_1. Then, Presenter plays the separation strategy for ω rounds in the region $\left[2M + 1 + \frac{5\varepsilon}{6}, 2M + 2 + \varepsilon\right]$ pushing to the left colors not in $\mathcal{X}_2$. Let Y_2 be the set of ω' left-most intervals from the second separation phase. Let $\mathcal{Y}_2$ denote the set of colors used by Algorithm to color Y_2. Let r be the left-most right endpoint of an interval in Y_1. Let l be the right-most left endpoint of an interval in Y_2.

There are two cases in the final phase. Let $\mathcal{C}_1 := \mathcal{X}_1 \cup \mathcal{Y}_1$, and analogously $\mathcal{C}_2 := \mathcal{X}_2 \cup \mathcal{Y}_2$. We have that $|\mathcal{C}_1| = |\mathcal{C}_2| = (\alpha + 1)\omega' - \delta = \omega - o(\omega)$.

Case 1. If $|\mathcal{C}_2 \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$, then Presenter introduces $\omega - \omega'$ times the same interval $[r, l]$. Each interval introduced in the final phase intersects with all intervals from both initial phases and all intervals in $Y_1 \cup Y_2$. Thus, Algorithm is forced to use $|\mathcal{C}_1 \cup \mathcal{C}_2| + \omega - \omega' = |\mathcal{C}_1| + |\mathcal{C}_2 \setminus \mathcal{C}_1| + \omega - \omega' \geqslant \omega - o(\omega) + \frac{\alpha + \frac{1}{2}}{\alpha+1}\omega = \left(2 - \frac{1}{2\alpha+2}\right)\omega - o(\omega)$ colors in total.

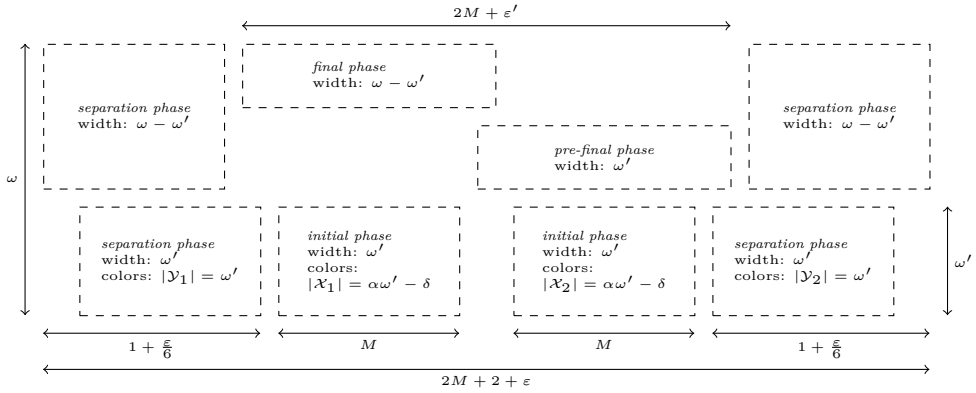


Figure 6 Lemma 17, Case 2: $|\mathcal{C}_2 \setminus \mathcal{C}_1| < \frac{\omega}{2\alpha+2}$.

Case 2. If $|\mathcal{C}_2 \setminus \mathcal{C}_1| < \frac{\omega}{2\alpha+2}$, then Presenter introduces ω' intervals, all of them having endpoints $[M + 1 + 5\varepsilon/12, l]$. Let Q be the set of colors used by Algorithm in this *pre-final phase*. We have $\mathcal{C}_2 \cap Q = \emptyset$, and we assumed that $|\mathcal{C}_2 \setminus \mathcal{C}_1| \leqslant \frac{\omega}{2\alpha+2}$, thus we have $|Q \setminus \mathcal{C}_1| \geqslant \frac{\omega}{2\alpha+2}$, and now we are in a situation analogous to Case 1, with Q playing the role of $\mathcal{C}_2$, see Figure 6.

The longest interval introduced by Presenter in both cases has length strictly less than $2M + \varepsilon$, and the whole game is played in the region $[0, 2M + 2 + \varepsilon]$. ◀

▶ **Corollary 18.** *There is an $\langle \alpha_n, 3 \cdot 2^n - 4 + \varepsilon, 3 \cdot 2^n - 2 + \varepsilon \rangle$-schema, for every $n \in \mathbb{N}_+$ and every $\varepsilon > 0$, where*

$$\alpha_n = \frac{(\sqrt{7}-4)(\sqrt{7}-3)^n + (\sqrt{7}+4)(-\sqrt{7}-3)^n}{(\sqrt{7}-1)(\sqrt{7}-3)^n + (\sqrt{7}+1)(-\sqrt{7}-3)^n}.$$

Proof. The argument is similar to Corollary 16, but now we solve the recurrence equations $\alpha_0 = 1$, $\alpha_{n+1} = 2 - \frac{1}{2\alpha_n+2}$, and $M_0 = 1$, $M_{n+1} = 2M_n + 2$, $\sigma_0 = 1, \sigma_{n+1} = 2M_n$. ◀

Note that, similarly to Observation 9, one could already use Corollary 18 to get a lower bound arbitrarily close to $\lim_{n\to\infty} \alpha_n = \frac{1+\sqrt{7}}{2} \approx 1.82287$ for the asymptotic competitive ratio of any online algorithm that work for all $\sigma \geqslant 1$. Nonetheless in Section 3.6 we prove a stronger $5/2$ lower bound.

▶ **Theorem 5.** *For every $\sigma > 2$ there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio less than $7/4$.*

Proof. Observe that, for $\sigma > 2$ and $n = 1$, Corollary 18 gives a $\langle \frac{7}{4}, 2 + (\sigma - 2), 4 + (\sigma - 2) \rangle$-schema. Then proceed analogously to the proof of Theorem 4. ◀

3.6 The $5/2$ Lower Bound

To prove our main negative result we need two simple combinatorial lemmas.

▶ **Lemma 19.** *Let $\gamma \in [0, 1]$. For every four sets $X_1, \ldots, X_4$, each of size k, if their intersection is small: $\left|\bigcap_{i=1}^4 X_i\right| \leqslant (1 - \gamma) \cdot k$, their union is large: $\left|\bigcup_{i=1}^4 X_i\right| \geqslant \frac{3+\gamma}{3} \cdot k$.*

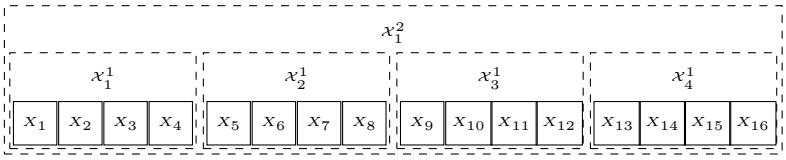


Figure 7 $\mathcal{X}_i^j$ sets in Lemma 20.

Proof. Each element which belongs to the union but does not belong to the intersection can belong to at most three sets. Thus, we have

$$3 \cdot \left(\left| \bigcup_{i=1}^{4} X_i \right| - \left| \bigcap_{i=1}^{4} X_i \right| \right) \geqslant 4 \cdot \left(k - \left| \bigcap_{i=1}^{4} X_i \right| \right),$$

and so

$$3 \cdot \left| \bigcup_{i=1}^{4} X_i \right| \geqslant 4k - \left| \bigcap_{i=1}^{4} X_i \right| \geqslant (3+\gamma) \cdot k. \qquad \blacktriangleleft$$

▶ **Lemma 20.** *Let $\gamma \in [0,1]$, and $X_1, \ldots, X_{4^n}$ be a family of 4^n sets, each of size k. Then, either*

$$\left| \bigcup_{i=1}^{4^n} X_i \right| \geqslant \left(\frac{3+\gamma}{3} \right)^n k,$$

or the sequence $1, 2, \ldots, 4^n$ can be covered with four disjoint intervals $[l_1, r_1], \ldots, [l_4, r_4]$, $l_1 = 1$, $l_{i+1} = r_i + 1$, $r_4 = 4^n$, such that for $Y_i = \bigcup_{j=l_i}^{r_i} X_j$ the intersection of Y_i's is large:

$$|Y_1 \cap Y_2 \cap Y_3 \cap Y_4| \geqslant (1-\gamma) \cdot k.$$

Proof. Consider $n+1$ families of sets defined as follows: $\mathcal{X}_i^0 := X_i$ for every $i \in [4^n]$, and $\mathcal{X}_i^j := \bigcup_{l=4i-3}^{4i} \mathcal{X}_l^{j-1}$ for every $j \in [n]$ and $i \in [4^{n-j}]$. See Figure 7.

If for some i, j we have $|\bigcap_{l=4i-3}^{4i} \mathcal{X}_l^j| \geqslant (1-\gamma) \cdot k$, then we are done. Thus, we assume that $\forall_{i,j}\ |\bigcap_{l=4i-3}^{4i} \mathcal{X}_l^j| < (1-\gamma) \cdot k$. Let $\varrho := \frac{3+\gamma}{3} \in \left[1, \frac{4}{3}\right]$. We prove that $\forall_{i,j}\ |\mathcal{X}_i^j| \geqslant \varrho^j k$, by induction on j. For $j = 0$ the statement is obvious because $\forall_i\ |\mathcal{X}_i^0| = |X_i| = k = \varrho^0 k$. For $j+1$ and arbitrary i, let $k' = \varrho^j k$. By the induction hypothesis $|\mathcal{X}_{4i-3}^j|, \ldots, |\mathcal{X}_{4i}^j| \geqslant \varrho^j k = k'$. We may ignore some elements of those sets and assume that $|\mathcal{X}_{4i-3}^j| = \ldots = |\mathcal{X}_{4i}^j| = k'$, moreover we assumed that $|\mathcal{X}_{4i-3}^j \cap \ldots \cap \mathcal{X}_{4i}^j| < (1-\gamma)k = \frac{1-\gamma}{\varrho^j} \varrho^j k = (1-\gamma')k'$, where $\gamma' \in [0,1]$ and $\gamma' > \gamma$. We apply Lemma 19 and get $|\mathcal{X}_{4i-3}^j \cup \ldots \cup \mathcal{X}_{4i}^j| \geqslant \frac{3+\gamma'}{3} k'$. Thus, $|\mathcal{X}_i^{j+1}| \geqslant \frac{3+\gamma'}{3} k' > \frac{3+\gamma}{3} k' = \varrho k' = \varrho^{j+1} k$. $\blacktriangleleft$

▶ **Lemma 21.** *If there is an $\langle \alpha, \sigma, M \rangle$-schema, then for every $\varepsilon > 0$ and for every $\gamma \in (0,1)$, there is a $\left\langle \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha, 4^n M + \varepsilon, 4^n M + \varepsilon \right\rangle$-schema, for some $n := n(\gamma)$.*

Proof. Let us fix an $\omega \in \mathbb{N}_+$, and let $\omega' = \left\lfloor \frac{\omega}{2} \right\rfloor$. Let S be an $\langle \omega', \alpha\omega' - \delta, \sigma, M \rangle$-strategy for some $\delta = o(\omega')$. Presenter repeats strategy S in the initial phase 4^n times. For each $i \in [4^n]$ the i-th game is played inside interval $\left[(i-1)(M + \frac{\varepsilon}{4^n}), (i-1)(M + \frac{\varepsilon}{4^n}) + M\right]$. See Figure 8. Algorithm uses $C = \alpha\omega' - \delta$ colors in each of these games. Let $\mathcal{X}_i$ denote the set of C colors used by Algorithm in the i-th game. Let $\mathcal{X}$ denote the set of all colors used in the initial phase, i.e., $\mathcal{X} = \bigcup_{i \in [4^n]} \mathcal{X}_i$.

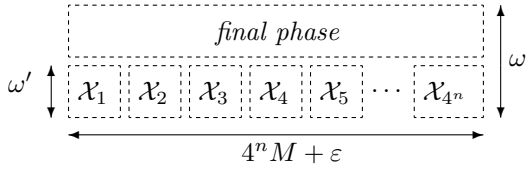


Figure 8 Case 1: $|\mathcal{X}|$ is large.

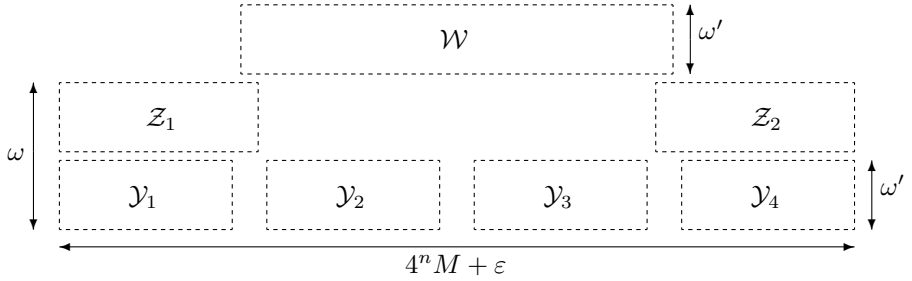


Figure 9 Case 2.1: $|\mathcal{Y}|$ is large and $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| \geqslant \frac{1}{4}\omega$.

We apply Lemma 20 to the family $\mathcal{X}_1, \ldots, \mathcal{X}_{4^n}$ and get that either the union of these sets has at least $\left(\frac{3+\gamma}{3}\right)^n C$ elements, or we get four disjoint consecutive subfamilies $\mathcal{Y}_1, \ldots, \mathcal{Y}_4$ ($\mathcal{Y}_i = \bigcup_{j=l_i}^{r_i} \mathcal{X}_j$) such that the size of the intersection $\mathcal{Y}_1 \cap \mathcal{Y}_2 \cap \mathcal{Y}_3 \cap \mathcal{Y}_4$ has at least $(1-\gamma) \cdot C$ elements.

Case 1. If the size of the union $|\mathcal{X}|$ is at least $\left(1 + \frac{\gamma}{3}\right)^n \cdot C$, then Presenter introduces ω' intervals, all of them having endpoints $[0, 4^n M + \varepsilon]$. See Figure 8. Each interval introduced in the final phase intersects with all intervals introduced in the initial phase. Thus, Algorithm is forced to use at least $|\mathcal{X}| + \omega' \geqslant \frac{1}{2}\left(\left(1 + \frac{\gamma}{3}\right)^n \alpha + 1\right)\omega - o(\omega)$ colors in total. Easy calculation shows that for $\gamma \in (0,1)$, $\alpha \in [1,3]$ and for any $n \geqslant \log_{1+\frac{\gamma}{3}}(5/2 - \gamma)$, we have $\frac{1}{2} + \frac{1}{2}\left(1 + \frac{\gamma}{3}\right)^n \alpha \geqslant \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha$.

Case 2. The size of the intersection $|\mathcal{Y}_1 \cap \ldots \cap \mathcal{Y}_4|$ is at least $(1-\gamma) \cdot C$. Let $\mathcal{Y} = \mathcal{Y}_1 \cap \mathcal{Y}_2 \cap \mathcal{Y}_3 \cap \mathcal{Y}_4$ denote the colors that appear in all four parts of the initial phase. Presenter introduces a set Z_1 of ω' identical intervals covering all intervals contributing to $\mathcal{Y}_1$ and disjoint with intervals contributing to $\mathcal{Y}_2$. See Figure 9. Let $\mathcal{Z}_1$ be the set of colors used by Algorithm to color Z_1.

Then Presenter introduces a set Z_2 of ω' identical intervals covering all intervals contributing to $\mathcal{Y}_4$ and disjoint with intervals contributing to $\mathcal{Y}_3$. Let $\mathcal{Z}_2$ be the set of colors used by Algorithm to color Z_2.

Clearly, $|\mathcal{Z}_1| = |\mathcal{Z}_2| = \omega'$, and $\mathcal{Z}_1 \cap \mathcal{Y} = \mathcal{Z}_2 \cap \mathcal{Y} = \emptyset$. Now we distinguish two subcases depending on the size of the set $\mathcal{Z}_2 \setminus \mathcal{Z}_1$.

Case 2.1. If $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| \geqslant \frac{1}{4}\omega$, then Presenter introduces a set W of ω' identical intervals intersecting all the intervals in Z_1 and Z_2, and covering all the intervals contributing to $\mathcal{Y}_2$ and $\mathcal{Y}_3$. Let $\mathcal{W}$ be the set of colors used by Algorithm to color W. By the definition, we have $\mathcal{W} \cap \mathcal{Y} = \mathcal{W} \cap \mathcal{Z}_1 = \mathcal{W} \cap \mathcal{Z}_2 = \emptyset$. Algorithm was forced to use $|\mathcal{W}| + |\mathcal{Z}_1| + |\mathcal{Z}_2 \setminus \mathcal{Z}_1| + |\mathcal{Y}| \geqslant \left(\frac{1}{2} + \frac{1}{2} + \frac{1}{4}\right)\omega + \frac{1}{2}(1-\gamma)\alpha\omega - o(\omega) = \left(\frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha\right)\omega - o(\omega)$ colors in total. See Figure 9.

Case 2.2. If $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| < \frac{1}{4}\omega$, then let $\mathcal{Z} = \mathcal{Z}_1 \cap \mathcal{Z}_2$ and observe that $|\mathcal{Z}| \geqslant \left\lfloor \frac{\omega}{4} \right\rfloor$. Presenter introduces a set W_1 of ω' identical intervals intersecting all the intervals in Z_1, and covering all the intervals contributing to $\mathcal{Y}_2$. Then, presenter introduces a set W_2 of

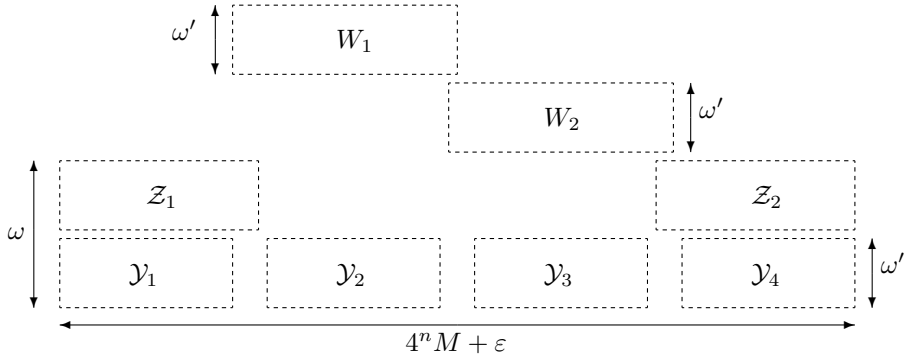


Figure 10 Case 2.2: $|\mathcal{Y}|$ is large and $|\mathcal{Z}_2 \setminus \mathcal{Z}_1| < \frac{1}{4}\omega$.

ω' identical intervals, intersecting all the intervals in W_1 and Z_2, and covering all the intervals contributing to $\mathcal{Y}_3$. Let $\mathcal{W}$ be the set of colors used by Algorithm to color intervals in $W_1 \cup W_2$. We have that $|\mathcal{W}| = 2\omega'$, and $\mathcal{W} \cap \mathcal{Y} = \mathcal{W} \cap \mathcal{Z} = \emptyset$. Algorithm was forced to use $|\mathcal{W}|+|\mathcal{Z}|+|\mathcal{Y}| \geqslant \left(1 + \frac{1}{4}\right)\omega + \frac{1}{2}(1-\gamma)\alpha\omega - o(\omega) = \left(\frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha\right)\omega - o(\omega)$ colors in total. See Figure 10. ◀

▶ **Corollary 22.** *There is an an $\langle \alpha_n, 4^{nf(\gamma)} + \varepsilon, 4^{nf(\gamma)} + \varepsilon \rangle$-schema, for every $n \in \mathbb{N}_+$, every $\varepsilon > 0$, and every $\gamma \in (0,1)$, where*

$$\alpha_n = \frac{5}{2} \cdot \frac{1}{1+\gamma} - \frac{(3-2\gamma)}{2(1+\gamma)} \cdot \left(\frac{1-\gamma}{2}\right)^n, \quad f(\gamma) = \left\lceil \frac{\log\left(\frac{5}{2} - \gamma\right)}{\log\left(1 + \frac{\gamma}{3}\right)} \right\rceil.$$

Proof. The argument is similar to Corollaries 16, and 18, but now we solve the recurrence equations $\alpha_0 = 1$, $\alpha_{n+1} = \frac{5}{4} + \frac{1}{2}(1-\gamma)\alpha_n$ for competitive ratio, and $M_0 = 1$, $M_{n+1} = 4^{f(\gamma)} M_n$, $\sigma_n = M_n$ for region and interval lengths. ◀

▶ **Theorem 6.** *For every $\varepsilon > 0$ there is $\sigma \geqslant 1$ such that there is no online algorithm for σ-interval coloring with the asymptotic competitive ratio $5/2 - \varepsilon$.*

Proof. Assume for contradiction that, for some $\varepsilon > 0$, there are $(5/2 - \varepsilon)$-competitive algorithms for every $\sigma \geqslant 1$. Setting γ small enough and n large enough, Corollary 22 gives us a $\left\langle \frac{5}{2} - \frac{\varepsilon}{4}, \sigma, \sigma \right\rangle$-schema, for some value of σ. This means, there is ω_P such that for every $\omega \geqslant \omega_P$ there exists an $\left\langle \omega, (\frac{5}{2} - \frac{2\varepsilon}{4})\omega, \sigma, \sigma \right\rangle$-strategy. On the other hand, for the assumed σ-interval coloring algorithm, there exists ω_A such that for every $\omega \geqslant \omega_A$ the algorithm uses at most $\left(\frac{5}{2} - \frac{3\varepsilon}{4}\right)\omega$ colors for every ω-colorable set of intervals. For $\omega = \max(\omega_A, \omega_P)$ we reach a contradiction. ◀

References

1 Yossi Azar, Amos Fiat, Meital Levy, and N.S. Narayanaswamy. An improved algorithm for online coloring of intervals with bandwidth. *Theoretical Computer Science*, 363(1):18–27, 2006. doi:10.1016/j.tcs.2006.06.014.

2 Seymour Benzer. On the topology of the genetic fine structure. *Proceedings of the National Academy of Sciences of the United States of America*, 45(11):1607–1620, 1959. doi:10.1073/pnas.45.11.1607.

3 Marek Chrobak and Maciej Ślusarek. On some packing problem related to dynamic storage allocation. *RAIRO, Theoretical Informatics and Applications*, 22(4):487–499, 1988. URL: http://www.numdam.org/item/ITA_1988__22_4_487_0.

4 Leah Epstein and Meital Levy. Online interval coloring and variants. In *ICALP 2005: 32nd International Colloquim on Automata, Languages and Programming, Lisbon, Portugal, July 2005. Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 602–613, 2005. doi:10.1007/11523468_49.

5 P. C. Fishburn and R. L. Graham. Classes of interval graphs under expanding length restrictions. *Journal of Graph Theory*, 9(4):459–472, 1985. doi:10.1002/jgt.3190090405.

6 Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs (Annals of Discrete Mathematics, Vol 57)*. Elsevier, 2 edition, 2004.

7 Magnús M. Halldórsson. Parallel and on-line graph coloring. *Journal of Algorithms*, 23(2):265–280, 1997. doi:10.1006/jagm.1996.0836.

8 Magnús M. Halldórsson and Mario Szegedy. Lower bounds for on-line graph coloring. *Theoretical Computer Science*, 130(1):163–174, 1994. doi:10.1016/0304-3975(94)90157-0.

9 Konstanty Junosza-Szaniawski, Paweł Rzążewski, Joanna Sokół, and Krzysztof Węsek. Online coloring and L(2,1)-labeling of unit disk intersection graphs. *SIAM Journal on Discrete Mathematics*, 32(2):1335–1350, 2018. doi:10.1137/16M1097821.

10 Henry A. Kierstead, David A. Smith, and William T. Trotter. First-fit coloring on interval graphs has performance ratio at least 5. *European Journal of Combinatorics*, 51:236–254, 2016. doi:10.1016/j.ejc.2015.05.015.

11 Henry A. Kierstead and William T. Trotter. An extremal problem in recursive combinatorics. In *12th Southeastern Conference on Combinatorics, Graph Theory and Computing, Baton Rouge, LA, USA, March 1981. Proceedings, vol. II*, volume 33 of *Congressus Numerantium*, pages 143–153, 1981. URL: https://people.math.gatech.edu/~trotter/papers/29.pdf.

12 C. Lekkeikerker and J. Boland. Representation of a finite graph by a set of intervals on the real line. *Fundamenta Mathematicae*, 51(1):45–64, 1962. URL: http://eudml.org/doc/213681.

13 N. S. Narayanaswamy and R. Subhash Babu. A note on first-fit coloring of interval graphs. *Order*, 25(1):49–53, 2008. doi:10.1007/s11083-008-9076-6.

14 Maciej Ślusarek. A coloring algorithm for interval graphs. In *MFCS 1989: Symposium on Mathematical Foundations of Computer Science, Porąbka-Kozubnik, Poland, August 1989. Proceedings*, volume 379 of *Lecture Notes in Computer Science*, pages 471–480, 1989. doi:10.1007/3-540-51486-4_93.

A Open Problems

Throughout the paper we presented several constructions which can be combined recursively to obtain strategies proving higher and higher lower bounds for larger and larger interval lengths σ. Table 1 summarizes a selection of these strategies.

Table 1 Summary of selected strategies for Presenter.

ratio	interval length	strategy
1.5	1	Epstein and Levy [4]
1.6	$2+\varepsilon$	Corollary 8, $n=2$ iterations
1.66	$1+\varepsilon$	Corollary 16, $n=1$ iteration
1.72	$3+\varepsilon$	Corollary 16, $n=2$ iterations
1.75	$2+\varepsilon$	Corollary 18, $n=1$ iteration
1.81	$8+\varepsilon$	Corollary 18, $n=2$ iterations
2	$4^{39}+\varepsilon$	Corollary 22, $n=3$ iterations, $\gamma=0.21030395$
2.4	$4^{486}+\varepsilon$	Corollary 22, $n=6$ iterations, $\gamma=0.0339$
2.49	$4^{7970}+\varepsilon$	Corollary 22, $n=10$ iterations, $\gamma=0.003449$

There are still large gaps between the best known lower and upper bounds for the optimal competitive ratios for online σ-interval coloring problems (see Figure 11). It would be interesting to close the gap, even for a single specific σ. For example, for $\sigma = 3/2$ the optimal online algorithm has the competitive ratio somewhere between $5/3$ and $5/2$.

Finally, let us conjecture that the lower bound of Theorem 6 is tight.

▶ **Conjecture 23.** *There is a $5/2$-competitive online algorithm for σ-interval coloring, for every $\sigma \geqslant 1$.*

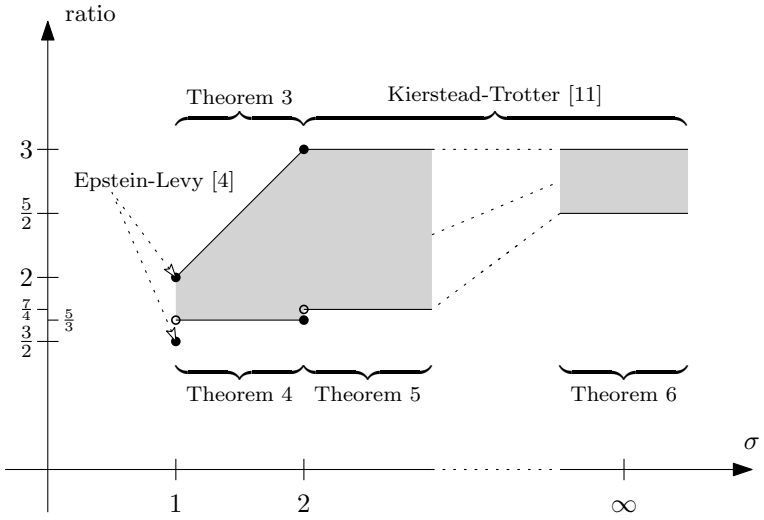


Figure 11 Gap between current bounds for competitive ratio of online σ-interval coloring.

Approximating Requirement Cut via a Configuration LP

Roy Schwartz
Department of Computer Science, Technion, Haifa, Israel
schwartz@cs.technion.ac.il

Yotam Sharoni
Department of Computer Science, Technion, Haifa, Israel
yotamsh@cs.technion.ac.il

Abstract

We consider the Requirement Cut problem, where given an undirected graph $G = (V, E)$ equipped with non-negative edge weights $c : E \to \mathbb{R}_+$, and g groups of vertices $X_1, \ldots, X_g \subseteq V$ each equipped with a requirement r_i, the goal is to find a collection of edges $F \subseteq E$, with total minimum weight, such that once F is removed from G in the resulting graph every X_i is broken into at least r_i connected components. Requirement Cut captures multiple classic cut problems in graphs, e.g., Multicut, Multiway Cut, Min k-Cut, Steiner k-Cut, Steiner Multicut, and Multi-Multiway Cut. Nagarajan and Ravi [Algoritmica'10] presented an approximation of $O(\log n \log R)$ for the problem, which was subsequently improved to $O(\log g \log k)$ by Gupta, Nagarajan and Ravi [Operations Research Letters'10] (here $R = \sum_{i=1}^{g} r_i$ and $k = |\cup_{i=1}^{g} X_i|$). We present an approximation of $O(X \log R \sqrt{\log k} \log \log k)$ for Requirement Cut (here $X = \max_{i=1,\ldots,g}\{|X_i|\}$). Our approximation in general is incomparable to the above mentioned previous results, however when all groups are not too large, i.e., $X = o\left((\sqrt{\log k} \log g)/(\log R \log \log k)\right)$, it is better. Our algorithm is based on a new configuration linear programming relaxation for the problem, which is accompanied by a remarkably simple randomized rounding procedure.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Mathematics of computing → Combinatorial optimization

Keywords and phrases Approximation, Requirement Cut, Sparsest Cut, Metric Embedding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.53

Category APPROX

Funding *Roy Schwartz*: Research is supported by ISF grant 1336/16.

1 Introduction

We consider the Requirement Cut problem (RC), where we are given an undirected graph $G = (V, E)$ equipped with non-negative edge weights $c : E \to \mathbb{R}_+$ and g groups of vertices $X_1, \ldots, X_g \subseteq V$. Every group X_i is associated with a requirement r_i, where $2 \leq r_i \leq |X_i|$. The goal is to find a collection of edges $F \subseteq E$, with total minimum weight, such that once F is removed from G in the resulting graph $G_F = (V, E \backslash F)$ every X_i is broken into at least r_i connected components. To simplify presentation we use the notation $\text{Cut}(G, X_i)$ to denote the number of connected components of G that contain at least one vertex of X. The above requirement implies that $\text{Cut}(G_F, X_i) \geq r_i$, $\forall i = 1, \ldots, g$.

Requirement Cut captures multiple classic cut problems in graphs, e.g., Multicut [11, 15], Multiway Cut [4, 5, 6, 9, 13, 22], Min k-Cut [19, 20, 21], Steiner k-Cut [8], Steiner Multicut [14], and Multi-Multiway Cut [3]. To simplify presentation of known results for (RC), we denote by R the sum of requirements, i.e., $R = \sum_{i=1}^{g} r_i$, by X the largest group, i.e., $X = \max_{i=1,\ldots,g}\{|X_i|\}$, and by k the number of vertices in groups, i.e., $k = |\cup_{i=1}^{g} X_i|$. Nagarajan and Ravi [18] were the first to consider (RC), presenting


© Roy Schwartz and Yotam Sharoni;
licensed under Creative Commons License CC-BY



approximations of $O(\log n \log R)$ and $O(\log R)$ for general graphs and trees, respectively.[1] They also proved that there is a hardness of approximation of $\Omega(\log g)$ when the graph is a tree. Subsequently, Gupta et al. [12] presented improved approximations of $O(\log k \log g)$ and $O(\log g)$ for general graphs and trees, respectively. Thus, providing a tight approximation when the graph is a tree. In both these works, the main approach to solving (RC) in general graphs is first to formulate a linear programming relaxation which finds a suitable spreading metric over the vertices of V, then transform the metric into a tree metric, and finally round the solution on the tree. In [18], an additional greedy combinatorial approach is presented. In this approach the algorithm repeatedly finds cuts that (approximately) minimize the ratio between their cost and the number of groups they separate. This algorithm also yields an approximation of $O(\log n \log R)$ for (RC).

1.1 Our Result

The following theorem summarizes our main result for the (RC) problem.

▶ **Theorem 1.** *There is a randomized polynomial time algorithm for* Requirement Cut *that achieves an approximation of* $O\left(X \log R \sqrt{\log k} \log\log k\right)$.

We note that our approximation is incomparable to the current state of the art [12]. However, when all groups are not too large, i.e., $X = o\left((\sqrt{\log k} \log g)/(\log R \log\log k)\right)$, our approximation is better. Our algorithm in fact provides an approximation guarantee of $O\left(DX \log R\right)$, where D is the distortion of embedding a metric of negative type into ℓ_1 (refer to Section 2 for an exact definition). The guarantee of Theorem 1 follows by employing the embedding of Arora et al. [2].

1.2 Our Techniques

Our approach for obtaining the result is based on three ingredients, on which we currently elaborate.

The first is a new configuration linear programming relaxation of exponential size, in which each cut S is associated with a variable λ_S. In this relaxation the goal is to assign a fractional value λ_S to each cut S while satisfying two requirements: (1) for each group X_i the total fractional value of cuts that separate it is high enough, i.e., X_i is broken into enough pieces; and (2) for each vertex u the total fractional value of cuts containing u is at most one, i.e., the connected components in the output are disjoint. This configuration LP differs from the relaxation used in [12, 18], which imposes a metric over the vertices of the graph while ensuring that for every X_i the total length of every tree spanning X_i is high enough.

The second ingredient is a remarkably simple randomized rounding procedure of the configuration LP. When considering classic randomized rounding applied to our setting, each cut S, independently of other cuts, chooses all edges that cross it with a probability of λ_S. This straightforward use of randomized rounding is problematic, as it might separate a group into very few pieces, even though the total fraction of cuts that separate the group is quite high. To intuitively exemplify this, it is enough to note that choosing ℓ cuts that separate some group X_i might result in breaking X_i into only $O(\log \ell)$ pieces due to non-trivial

[1] In [18], in contrast to what is cited above, the claimed approximation does not depend on R but rather on the maximum requirement multiplied by g. However, the algorithms of [18] provide the above claimed approximation guarantee.

overlaps between the cuts. To overcome this difficulty, we note that cuts S with a high fractional value λ_S are disjoint. Hence, for these cuts the above difficulty does not occur. Unfortunately, cuts S with a high fractional value λ_S might be few and we are not guaranteed that choosing all of them breaks each X_i into enough pieces. To remedy this, we introduce a remarkably simple analysis for the classic randomized rounding procedure that assumes all cuts S have a small fractional value λ_S. In our analysis we show that for each cut S, if all other cuts have a small fractional value, then with a high enough probability S is able to break at least one additional piece from X_i on its own. This enables us to lower bound the number of pieces each X_i is broken into.

The third and last ingredient relates to solving the configuration LP. Since the configuration LP has an exponential number of variables, we need to present an (approximate) dual separation oracle for it. It turns out that the dual separation oracle for the configuration LP is a node weighted variant of Sparsest Cut with general demands where demands are given over groups, as opposed to pairs. Our approximation algorithm for the dual separation oracle, similarly to classic algorithms for Sparsest Cut, finds a suitable metric of negative type, embeds it into ℓ_1, and chooses the best cut S among all cuts in the decomposition of the ℓ_1 metric into a non-negative combination of cut metrics. The only caveat in this approach is that, unlike Sparsest Cut, our problem is not symmetric, i.e., S and $\overline{S}$ have different objective value. To overcome this, we break symmetry and introduce an artificial point to the metric space, while ensuring that all cuts in the decomposition of the ℓ_1 metric do not contain this artificial point.

We note that our dual separation oracle captures the Steiner Ratio problem (SR) [14], used in the greedy step of the combinatorial approach of [18] for (RC). Thus, it can be proved that our approximation guarantee of $O(X\sqrt{\log k}\log\log k)$ for the dual separation oracle (Theorem 8) also extends to (SR). Incorporating this into the analysis of [18] yields an overall approximation of $O\left(X\log R\sqrt{\log k}\log\log k\right)$, matching the result of Theorem 1 in yet another way.

1.3 Related Work

The literature on approximating cut problems and metric embedding is vast, we mention here only some of the most relevant work. Let us start by surveying problems captured by (RC). Multicut, a special case of (RC) where each group X_i is of size two and its requirement r_i also equals two, admits an approximation of $O(\log C\log^2 k)$ (where C is the total weight of edges in the graph) [15] and $O(\log k)$ [11]. Multiway Cut, another special case of (RC) where there is a single group X of size k and its requirement also equals k, exhibits a long sequence of works [4, 5, 6, 9, 13, 22] which culminates in an approximation of 1.2965 [22]. Min k-Cut, a special case of (RC) where there is a single group X that spans all of V and its requirement equals k, admits a simple combinatorial approximation of $2(1-1/k)$ [19, 20, 21]. Steiner k-Cut is similar to Min k-Cut, with the difference that X does not necessarily equal all of V, also admits a similar approximation of $2(1-1/k)$ [8]. Steiner Multicut is similar to Multicut, with the difference that each group X_i might be larger than two, though its requirement r_i still remains two. This problem admits an approximation of $O(\log^3(gX))$ [14], which was subsequently improved to $O(\log g\log n)$ [18] and to $O(\log g\log k)$ [12]. Multi-Multiway Cut is similar to Multiway Cut, with the difference that there are multiple groups (though the requirement of each group equals its size). This problem admits an approximation of $O(\log g)$ [3].

Metric embedding also play a central role in our work, specifically embedding metrics of negative type into ℓ_1. Chawla et al. [7] presented an embedding with a distortion of $O(\log^{3/4} n)$, which was later improved to $O(\sqrt{\log n}\log\log n)$ by Arora et al. [1, 2]. The above

works are based on the seminal result of Arora et al. for Sparsest Cut, which obtained an approximation of $O(\sqrt{\log n})$, improving upon the $O(\log n)$ guarantee of Leighton and Rao [16].

Paper Organization

Section 2 contains needed preliminaries regarding metric spaces. Section 3 presents our new configuration LP for (RC). The rounding algorithm appears in Section 4, and Section 5 is dedicated to approximately solving the configuration LP. All missing proofs appear in the appendix.

2 Preliminaries

A metric space (V, d) embeds into ℓ_1 with distortion D if there exists an embedding f satisfying: $d(a,b) \leq ||f(a) - f(b)||_1 \leq D \cdot d(a,b)$, $\forall a, b \in V$. Moreover, a metric space (V, d) is of negative-type if $(V, \sqrt{d})$ is isometric to a subset of Euclidean space. Arora et al. [1, 2] proved that $D = O(\sqrt{\log n} \log\log n)$ for metrics of negative type, when $|V| = n$. We can exploit a slightly improved guarantee (see Corollary 5.1 in [2]), which we rephrase here for simplicity of presentation.[2]

▶ **Theorem 2** (Corollary 5.1 in [2]). *Given a metric space (V, d) of negative type and $U \subseteq V$ of size k, there exists an embedding f into ℓ_1 that satisfies: (1) $||f(a) - f(b)||_1 \leq D \cdot d(a,b)$, $\forall a, b \in V$; and (2) $||f(a) - f(b)||_1 \geq d(a,b)$, $\forall a, b \in U$. In the above $D = O(\sqrt{\log k} \log\log k)$. Furthermore, f can be computed in polynomial time.*

We note that in the above theorem, expansion is upper bounded for all pairs of points in the metric space, whereas contraction is lower bounded only for pairs of points in the given subset. This enables the improvement in the value of D from $O(\sqrt{\log n} \log\log n)$ to $O(\sqrt{\log k} \log\log k)$.

3 Configuration LP

We consider the following new linear relaxation for the (RC) problem. In this relaxation each possible cut $S \subseteq V$ is associated with a variable λ_S. We denote by $\delta_G(S)$ the total weight of edges crossing the cut S defines in the graph G, and let $\mathcal{C}(T) = \{S : 0 < |S \cap T| < |T|\}$ be the collection of cuts S that separate T, i.e., S contains at least one vertex of T but does not contain T as a whole.

$$
\begin{aligned}
\text{(LP)} \quad \min \quad & \sum_{S \in V} \frac{1}{2} \delta_G(S) \cdot \lambda_S & & \\
\text{s.t.} \quad & \sum_{S \in \mathcal{C}(X_i)} \lambda_S \geq r_i & \forall i = 1, \ldots g & \quad (1) \\
& \sum_{S : u \in S} \lambda_S \leq 1 & \forall u \in V & \quad (2) \\
& \lambda_S \geq 0 & \forall S \subseteq V &
\end{aligned}
$$

In the above relaxation, Constraint (1) ensures that X_i is broken into at least r_i pieces, and Constraint (2) ensures that each vertex belongs to at most a single cut.

[2] The cited theorem of [2] is originally given for ℓ_2 and not ℓ_1. However, standard arguments imply it also applies to ℓ_1, see, e.g., [2], for further details.

For our rounding algorithm to work we actually need to iteratively resolve (LP) for subgraphs of G. To this end, let us formally define the relaxation when considering $G_F = (V, E \setminus F)$, for some $F \subseteq E$, which we denote by (LP$_F$). Moreover, we denote by $Y_{i,1}, \ldots, Y_{i,\ell_i}$ the ℓ_i pieces X_i is broken into in G_F. Formally, if G_F is broken into ℓ connected components $C_1, \ldots, C_\ell$, which form a partition of V, then every $Y_{i,j}$ is obtained by $X_i \cap C_j$ and discarding the result if X_i and C_j are disjoint (hence $\ell_i \leq \ell$). Additionally, let us denote by r'_i the residual requirement of X_i in G_F, i.e., $r'_i = \max\{0, r_i - \ell_i\}$ (we can assume that groups X_i for which the residual requirement reached zero are removed from the instance).

$$
\begin{aligned}
(\mathrm{LP}_F) \quad \min \quad & \sum_{S \in V} \frac{1}{2} \delta_{G_F}(S) \cdot \lambda_S & & \\
\text{s.t.} \quad & \sum_{j=1}^{\ell_i} \sum_{S \in C(Y_{i,j})} \lambda_S \geq r'_i & \forall i = 1, \ldots g & \quad (3) \\
& \sum_{S : u \in S} \lambda_S \leq 1 & \forall u \in V & \quad (4) \\
& \lambda_S \geq 0 & \forall S \subseteq V &
\end{aligned}
$$

Similarly to (LP), Constraint (3) ensures that X_i is broken to at least r'_i pieces, and Constraint (4) ensures that each vertex belongs to at most a single cut.

The following lemma proves that (LP$_F$), for every $F \subseteq E$, is a relaxation when considering G_F. We denote by $\mathrm{OPT}_{\mathrm{LP}_F}$ the value of an optimal solution to (LP$_F$) and by OPT the value of an optimal solution to the original instance.

▶ **Lemma 3.** *$OPT_{LP_F} \leq OPT$, for every $F \subseteq E$.*

Proof. Let F^* be an optimal solution to the problem with respect to the original instance, and denote by $\tilde{F} = F^* \setminus F$ the edges of F^* still remaining in G_F. Thus, $G_{\tilde{F} \cup F} = (V, E \setminus (\tilde{F} \cup F))$ denotes the graph resulting in the removal of F^* from G_F. Let $S^*_1, \ldots, S^*_\ell$ be the connected components in $G_{\tilde{F} \cup F}$. Define the following solution to (LP$_F$): $\lambda_{S^*_i} = 1$ for every $i = 1, \ldots, \ell$ (and $\lambda_S = 0$ for all other cuts S). We notice that $S^*_1, \ldots, S^*_\ell$ are disjoint cuts, hence Constraint (4) is satisfied for every $u \in V$. Since F^* is a feasible solution with respect to the original instance, i.e., X_i is broken to at least r_i pieces in G_{F^*} (or equivalently $\mathrm{Cut}(G_{F^*}, X_i) \geq r_i$), we can conclude that removing $\tilde{F}$ from G_F breaks X_i to at least the same number of pieces. Thus, since X_i is already broken into ℓ_i pieces in G_F: $Y_{i,1}, \ldots, Y_{i,\ell_i}$, we can conclude that $\sum_{j=1}^{\ell_i} \mathrm{Cut}(G_{\tilde{F} \cup F}, Y_{i,j}) \geq r'_i$. Therefore, Constraint (3) is also satisfied and the defined solution is feasible for (LP$_F$). Since $\delta_{G_F}(S) \leq \delta_G(S)$, for every $S \subseteq V$, we can conclude that the value of the solution is at most OPT. This finishes the proof. ◀

A main challenge is solving (LP$_F$) since it has an exponential number of variables. The following theorem summarizes our guarantee for solving (LP$_F$), and Section 5 is dedicated to its proof. The solution found is bicriteria as it violates the constraints and incurs some loss in the objective.

▶ **Theorem 4.** *For every $F \subseteq E$, there exists an efficient algorithm that finds a bicriteria solution $\{\lambda_S\}_{S \subseteq V} \subseteq \mathbb{R}_+$ to (LP$_F$) satisfying:*

1. $\sum_{S : u \in S} \lambda_S \leq \alpha$, $\forall u \in V$.
2. $\sum_{j=1}^{\ell_i} \sum_{S \in C(Y_{i,j})} \lambda_S \geq \beta \cdot r'_i$, $\forall i = 1, \ldots, g$.
3. $\sum_{S \subseteq V} \frac{1}{2} \delta_{G_F}(S) \lambda_S \leq \gamma \cdot OPT$.

In the above $\alpha = \gamma = O(XD)$ (where D is as in Theorem 2 when considering $U = \cup_{i=1}^{g} X_i \subseteq V$) and $\beta = 1$.

4 Rounding the Configuration LP

In this section we present our rounding algorithm and prove the main result, Theorem 1, given Theorem 4. Our rounding algorithm progresses in iterations, similar in spirit to the classic randomized rounding for the Set Cover problem (a similar method was employed in the context of (RC) by [18]). However, unlike the latter rounding method, we require a different handling of cuts S with a large λ_S fraction and cuts S with a small λ_S fraction.

We start by focusing on the algorithm for a single iteration, which given G_F, for some $F \subseteq E$, outputs a random collection $L \subseteq E \setminus F$ of edges such that: (1) if L is removed from G_F then the number of pieces every X_i is broken into increases additively in expectation, up to some scaling, by X_i's residual requirement r'_i; and (2) the expected cost of edges in L is not too large. We denote by $\{\tilde{\lambda}_S\}_{S\subseteq V}$ the solution $\{\lambda_S\}_{S\subseteq V}$ scaled by a factor of $1/\alpha$, where α is as in guarantee (1) of Theorem 4, *i.e.*, $\tilde{\lambda}_S = \lambda_S/\alpha$. The single iteration algorithm removes all edges crossing a collection of cuts chosen according to two different criterions to ensure that the number of pieces X_i is broken into increases as required. The first criterion consists of all cuts whose $\tilde{\lambda}_S$ value is sufficiently large. The second criterion consists of executing randomized rounding on all cuts whose $\tilde{\lambda}_S$ is sufficiently small. Algorithm 1 summarizes the single iteration rounding procedure (in the algorithm's description we denote by $\Gamma_G(S)$ the collection of edges in the graph G crossing the cut S), and Lemma 5 summarizes the guarantee of Algorithm 1.

Algorithm 1 Single Iteration.

Input: $G_F = (V, E\setminus F), \{\tilde{\lambda}_S\}_{S\subseteq V}$
Output: $L \subseteq E\setminus F$
$F_1 \leftarrow \{e \in E \setminus F : \exists S, \tilde{\lambda}_S \geq 2/3 \wedge e \in \Gamma_{G_F}(S)\}$
let $\{I_S\}_{S\subseteq V:\tilde{\lambda}_S<2/3}$ be independent random indicators where $\Pr[I_S = 1] = \tilde{\lambda}_S$
$F_2 \leftarrow \{e \in E \setminus F : \exists S \text{ s.t. } \tilde{\lambda}_S < 2/3 \wedge I_S = 1 \wedge e \in \Gamma_{G_F}(S)\}$
$L \leftarrow F_1 \cup F_2$
output L

▶ **Lemma 5.** *For every $F \subseteq E$ and $\{\lambda_S\}_{S\subseteq V}$ guaranteed in Theorem 4, executing Algorithm 1 on G_F with $\{\tilde{\lambda}_S\}_{S\subseteq V}$ results in $L \subseteq E \setminus F$ satisfying:*
1. $\mathbb{E}\left[Cut(G_{F\cup L}, X_i)\right] \geq Cut(G_F, X_i) + \Omega\left(\beta/\alpha\right) \cdot r'_i$, $\forall i = 1, ...g$.
2. $\mathbb{E}\left[\sum_{e\in L} c_e\right] \leq O\left(\gamma/\alpha\right) \cdot OPT$.

Proof. Let us start with requirement (1) above. Note that for every vertex $u \in V$, since $\{\lambda_S\}_{S\subseteq V}$ satisfies guarantee (1) of Theorem 4, we can conclude that: $\sum_{S:u\in S} \tilde{\lambda}_S \leq 1$. Thus, we can conclude that all cuts S whose $\tilde{\lambda}_S$ is large, i.e., $\tilde{\lambda}_S \geq 2/3$, are disjoint. Therefore, removing F_1 from G_F additively increases the number of pieces every X_i is broken into by:

$$\sum_{j=1}^{\ell_i} \left|\left\{S : S \in C(Y_{i,j}) \wedge \tilde{\lambda}_s \geq 2/3\right\}\right| \geq \sum_{j=1}^{\ell_i} \sum_{S:S\in C(Y_{i,j})\wedge\tilde{\lambda}_S\geq 2/3} \tilde{\lambda}_S,$$

where the above inequality follows from the fact that $\tilde{\lambda}_S \leq 1, \forall S \subseteq V$.

Let us now consider cuts $S \subseteq V$ whose $\tilde{\lambda}_S$ is small, i.e., $\tilde{\lambda}_S < 2/3$. We analyze the expected additive increase in the number of pieces X_i is broken into when removing F_2 from G_F. A cut S satisfying: $\tilde{\lambda}_S < 2/3$ and $S \in C(Y_{i,j})$, increases the number of pieces $Y_{i,j}$ is broken into by one if there exists a vertex $u \in Y_{i,j} \cap S$ such that S is the only cut having $I_S = 1$, among all cuts T with $\tilde{\lambda}_T < 2/3$ containing u. Let us denote this event by $A_{Y_{i,j},S,u}$. Thus,

$$\Pr\left[A_{Y_{i,j},S,u}\right] = \tilde{\lambda}_S \cdot \prod_{T:\tilde{\lambda}_T<2/3\wedge u\in T} \left(1 - \tilde{\lambda}_T\right).$$

We note that since $\sum_{T:u\in T}\tilde{\lambda}_T \leq 1$ (as previously mentioned in the proof), and the fact that we consider only cuts T satisfying $\tilde{\lambda}_T < 2/3$, the following holds: $\prod_{T:\tilde{\lambda}_T<2/3\wedge u\in T}(1-\tilde{\lambda}_T) = \Omega(1)$. Hence, $\Pr[A_{Y_{i,j},S,u}] \geq \Omega(\tilde{\lambda}_S)$. Therefore, removing F_2 from G_F additively increases the expected number of pieces every X_i is broken into by at least:

$$\sum_{j=1}^{\ell_i}\sum_{S:S\in C(Y_{i,j})\wedge\tilde{\lambda}_S<2/3}\Pr\left[\exists u\in Y_{i,j}\cap S \text{ s.t. } A_{Y_{i,j},S,u}\right] \geq \sum_{j=1}^{\ell_i}\sum_{S:S\in C(Y_{i,j})\wedge\tilde{\lambda}_S<2/3}\Omega(\tilde{\lambda}_S).$$

The above inequality follows since for every $Y_{i,j}$ and every $S\in C(Y_{i,j})$, satisfying $\tilde{\lambda}_S < 2/3$, one can choose an arbitrary vertex $u\in Y_{i,j}\cap S$ and apply the lower bound on the probability of the event $A_{Y_{i,j},S,u}$.

Recall that $\{\lambda_S\}_{S\subseteq V}$ satisfies guarantee (2) of Theorem 4, therefore we can conclude that $\forall i = 1,\ldots,g$:

$$\sum_{j=1}^{\ell_i}\sum_{S:S\in C(Y_{i,j})\wedge\tilde{\lambda}_S<2/3}\tilde{\lambda}_S \geq \frac{\beta}{2\alpha}r'_i \qquad \text{or} \qquad \sum_{j=1}^{\ell_i}\sum_{S:S\in C(Y_{i,j})\wedge\tilde{\lambda}_S\geq 2/3}\tilde{\lambda}_S \geq \frac{\beta}{2\alpha}r'_i.$$

Thus, removing all edges in $F_1\cup F_2$ from G_F additively increases the number of pieces every X_i is broken into, in expectation, by at least $\Omega\left(\beta/\alpha\right)r'_i$. This concludes the proof of requirement (1).

Let us focus on requirement (2). It is easy to note that the definition of F_1 implies that $\sum_{e\in F_1}c_e \leq 3/2\sum_{S:\tilde{\lambda}_S\geq 2/3}\delta_{G_F}(S)\tilde{\lambda}_S$. Additionally, it is easy to note from the definition of F_2 that $\mathbb{E}[\sum_{e\in F_2}c_e] \leq \sum_{S:\tilde{\lambda}_S<2/3}\delta_{G_F}(S)\tilde{\lambda}_S$. Summing the former and the latter and recalling that $\tilde{\lambda}_S = \lambda_S/\alpha\ \forall S\subseteq V$, along with the fact that $\{\lambda_S\}_{S\subseteq V}$ satisfies guarantee (3) of Theorem 4, we can conclude that: $\mathbb{E}[\sum_{e\in L}c_e] \leq O(\gamma/\alpha)\mathrm{OPT}$. This concludes the proof of requirement (2). ◀

We are now ready to describe our rounding algorithm, which is depicted in Algorithm 2. The algorithm is straightforward, as it iteratively applies the single iteration algorithm, Algorithm 1, as long as there is a group X_i that is not broken to its required r_i number of pieces. We note that unlike previous algorithms for the (RC) problem, e.g., [18], we need to resolve (LP_F) in each iteration as it is not clear whether the original fractional solution remains feasible for G_F.

Algorithm 2 Rounding.

Input: $G(V,E), \{X_i\}_{i=1}^g, \{r_i\}_{i=1}^g, c: E\to\mathbb{R}_+$
Output: $F\subseteq E$

```
F ← ∅
while ∃i = 1,...,g s.t. X_i is not broken into at least r_i pieces in G_F do
    solve (LP_F) to obtain {λ_S}_{S⊆V} using Theorem 4
    λ̃_S ← λ_S/α, ∀S ⊆ V
    execute Algorithm 1 with G_F and {λ̃_S}_{S⊆V} to obtain L
    F ← F ∪ L
output F
```

The following lemma bounds the number of iterations Algorithm 2 performs, and its proof follows similar lines to the proof of [18]. We present it here for completeness.

▶ **Lemma 6.** *With a probability of at least* $1/2$ *Algorithm 2 performs at most* $O(\alpha/\beta \cdot \log R)$ *iterations.*

Proof. Let R_i^s be the random variable that equals the residual requirement of X_i at the beginning of iteration s of Algorithm 2, and let $R^s = \sum_{i=1}^{g} R_i^s$ be the random variable that equals the total residual requirement at the beginning of iteration s. Requirement (1) of Lemma 5 implies that there exists an absolute constant c such that for every iteration s: $\mathbb{E}[R_i^s | R_i^{s-1}] \leq (1 - c \cdot \beta/\alpha) R_i^{s-1}$, where $(1 - c \cdot \beta/\alpha) < 1$. From linearity of expectation we get that for every iteration s: $\mathbb{E}[R^s] \leq (1 - c \cdot \beta/\alpha)^s R$. Recall that R denotes the total initial requirement, i.e., $R = R^0 = \sum_{i=1}^{g} r_i$. Thus, if we choose $s^* = \log(4R) / \log((1 - c \cdot \beta/\alpha)^{-1})$ then $\mathbb{E}[R^{s^*}] \leq 1/4$. Using Markov's Inequality we get that $\Pr[R^{s^*} < 1/2] \geq 1/2$. However, since R^{s^*} is integral we can conclude that with a probability of at least $1/2$ Algorithm 2 terminates after s^* iterations because all requirements are satisfied, i.e., R^{s^*} reached a value of 0. Noting that $s^* = O(\alpha/\beta \cdot \log R)$ concludes the proof. ◀

Proof of Theorem 1. Lemma 6 proves that with a probability of at least $1/2$ Algorithm 2 performs $O(\alpha/\beta \cdot \log R)$ iterations. Requirement (2) of Lemma 5 implies that the expected cost of every iteration is at most $O(\gamma/\alpha) \cdot \mathrm{OPT}$. Hence, the expected value of the output of the algorithm is $O(\gamma/\beta \cdot \log R) \cdot \mathrm{OPT}$. Plugging the values of γ and β as guaranteed in Theorem 4 concludes the proof. ◀

5 Solving the Configuration LP

In this section we address the problem of solving the relaxation (LP_F). This task requires us to provide an (approximate) separation oracle for the dual of (LP_F), thus proving Theorem 4. Intuitively, the dual separation oracle is a node weighted variant of sparsest cut where demands are given over subsets, as opposed to pairs, of vertices. We denote this problem by Sparsest Requirement Cut (SRC). There are multiple methods of proving that an approximate dual separation oracle provides a bicriteria solution to the primal formulation, e.g., via the Ellipsoid algorithm. In this version of the paper we use Young's iterative method [23]. To simplify presentation, let us denote by $\mathrm{Cross}(S, i) = |\{Y_{i,j} : S \in C(Y_{i,j})\}|$ the number of sets among $Y_{i,1}, \ldots, Y_{i,\ell_i}$ that S separates.

5.1 The Sparsest Requirement Cut Problem

Let us start by formally introducing the (SRC) problem, which is essential to proving Theorem 4.

▶ **Definition 7.** *An instance of the Sparsest Requirement Cut problem consists of the following tuple* $(G, F, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c, y, z)$*, where:* $G = (V, E)$ *is an undirected graph,* $F \subseteq E$ *is a collection of edges removed from* G*, for every* $i = 1, \ldots, g$*:* $\{Y_{i,j}\}_{j=1}^{\ell_i}$ *is the partition of* X_i *according to the connected components of* G_F*, non-negative edge weights* $c : E \to \mathbb{R}_+$*, non-negative group weights* $z : \{1, \ldots, g\} \to \mathbb{R}_+$*, and non-negative vertex weights* $y : V \to \mathbb{R}_+$*. The goal is to find a cut* $S \subseteq V$ *minimizing:*

$$\frac{\delta_{G_F}(S) + \sum_{u \in S} y_u}{\sum_{i=1}^{g} \mathit{Cross}(S, i) z_i}.$$

The following theorem summarizes our algorithm for (approximately) solving (SRC), and it has a key role in proving Theorem 4. Its proof follows the lines of the classic algorithm for Sparsest Cut (SC) with general demands (see, e.g., [17]): first a metric is found that forms a lower bound on the value of an optimal solution, then it is rounded by embedding it

into ℓ_1. Unfortunately, (SRC) is inherently different from (SC) since it is not symmetric, i.e., S and $\overline{S}$ have (possibly) different objective values. To overcome this we introduce a new artificial point o to the metric space which represents $\overline{S}$. Furthermore, when embedding the metric into ℓ_1 we make sure to consider only cuts that do not contain o.

▶ **Theorem 8.** *Given an instance $(G, F, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c, y, z)$ of the Sparsest Requirement Cut problem, there exists a polynomial time algorithm that returns a cut S satisfying:*

$$\frac{\delta_{G_F}(S) + \sum_{u\in S} y_u}{\sum_{i=1}^{g} Cross(S,i)z_i} \leq (X-1)\cdot D \cdot OPT_{SRC},$$

where OPT_{SRC} is the value of an optimal solution to the given instance, and D is as in Theorem 2 when considering $U = \cup_{i=1}^{g} X_i \subseteq V$.

Proof. Our proof is comprised of two steps: (1) introducing a semi-definite relaxation and proving it lower bounds $\mathrm{OPT}_{\mathrm{SRC}}$; and (2) rounding the fractional solution and proving it yields the desired approximation factor.

Step 1. We start by presenting a semi-definite relaxation for (SRC) which we denote by (SDP). In (SDP), a squared Euclidean metric space is imposed over $V \cup \{o\}$, where o denotes a special artificial point which is the origin. Thus, every vertex $a \in V \cup \{o\}$ is associated with a vector $\mathbf{v}_a$ and $\mathbf{v}_o$ is constrained to be the origin, i.e., the zero vector. Moreover, we denote by $\mathcal{T}_{i,j}$ the collection of all spanning trees over the complete graph whose vertices are $Y_{i,j}$.

$$\begin{aligned}
\min \quad & \sum_{e=(a,b)\in E\setminus F} c_e ||\mathbf{v}_a - \mathbf{v}_b||_2^2 + \sum_{a\in V} y_a ||\mathbf{v}_a||_2^2 & & \\
\text{s.t.} \quad & ||\mathbf{v}_a - \mathbf{v}_b||_2^2 + ||\mathbf{v}_b - \mathbf{v}_c||_2^2 \geq ||\mathbf{v}_a - \mathbf{v}_c||_2^2 & \forall a,b,c \in V\cup\{o\} & \quad (5)\\
& ||\mathbf{v}_o||_2^2 = 0 & & \quad (6)\\
& \sum_{(a,b)\in T} ||\mathbf{v}_a - \mathbf{v}_b||_2^2 \geq s_{i,j} & \forall i = 1,\ldots,g, \forall j = 1,\ldots,\ell_i, \forall T \in \mathcal{T}_{i,j} & \quad (7)\\
& \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} s_{i,j} \geq 1 & & \quad (8)
\end{aligned}$$

Constraint (5) is the triangle inequality, whereas Constraint (6) enforces that $\mathbf{v}_o$ is the origin. Constraint (7) states that each $s_{i,j}$ is upper bounded by the length of the minimum spanning tree in $\mathcal{T}_{i,j}$. Constraint (8) is a scaling constraint, similar to the standard relaxation for (SC). Clearly, (SDP) is solvable in polynomial time since the separation oracle is just the computation of a minimum spanning tree.

Let us now prove that the optimal value of (SDP) lower bounds $\mathrm{OPT}_{\mathrm{SRC}}$. Let S^* be an optimal solution to the problem, let us define the following solution to (SDP):

$$\mathbf{v}_a = \begin{cases} 0 & a \notin S^* \\ \frac{1}{\sqrt{\sum_{i=1}^{g} \mathrm{Cross}(S^*,i)z_i}}\mathbf{e} & a \in S^* \end{cases} \quad \text{and} \quad s_{i,j} = \begin{cases} \frac{1}{\sum_{i=1}^{g} \mathrm{Cross}(S^*,i)z_i} & S^* \in C(Y_{i,j}) \\ 0 & S^* \notin C(Y_{i,j}) \end{cases},$$

where $\mathbf{e}$ is an arbitrary unit vector. Additionally, we set $\mathbf{v}_0 = \mathbf{0}$. The crucial observation is that $||\mathbf{v}_a - \mathbf{v}_b||_2^2 = (\sum_{i=1}^{g} \mathrm{Cross}(S^*,i)z_i)^{-1}$ if a and b are on different sides of S^* and 0 otherwise. Clearly, the above solution satisfies Constraints (5) and (6). Focusing on Constraint (7), if $s_{i,j} \neq 0$ then $S^* \in C(Y_{i,j})$ and the minimum spanning tree of $Y_{i,j}$ crosses S^* exactly once and its length equals $(\sum_{i=1}^{g} \mathrm{Cross}(S^*,i)z_i)^{-1}\mathbf{e}\cdot\mathbf{e} = s_{i,j}$. Considering Constraint (8), we note that:

$$\sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} s_{i,j} = \sum_{i=1}^{g} z_i \frac{\mathrm{Cross}(S^*,i)}{\sum_{i=1}^{g} \mathrm{Cross}(S^*,i)z_i} = 1.$$

Hence, we can conclude that the above solution is feasible. Moreover, we note that the objective value of the above defined solution equals:

$$\sum_{e=(a,b)\in E\setminus F} c_e||\mathbf{v}_a-\mathbf{v}_b||_2^2+\sum_{a\in V} y_a||\mathbf{v}_a||_2^2=\frac{\delta_{G_F}(S^*)+\sum_{u\in S^*}y_u}{\sum_{i=1}^{g}\text{Cross}(S^*,i)z_i}.$$

Therefore, the value of an optimal solution to (SDP) lower bounds OPT_{SRC}.

Step 2. We start by presenting the rounding algorithm, Algorithm 3. As previously mentioned, Algorithm 3 follows the lines of classis algorithms for (SC) with non-uniform demands [17]. The main difference is that we make sure to consider only cuts that do not contain o when decomposing the ℓ_1 metric into a non-negative combination of cut metrics (such a decomposition can be easily found, see, e.g., [10, 17]). In what follows we use the notation δ_S to denote the cut metric S defines, i.e., $\delta_S(a,b)=1$ if a and b are on different sides of S and 0 otherwise.

Algorithm 3 Rounding (SDP).

Input: $\{\mathbf{v}_a\}_{a\in V\cup\{o\}}$
Output: $S\subseteq V$
let f be the embedding into ℓ_1 of Theorem 2 when considering $U=\cup_{i=1}^{g}X_i\subseteq V\cup\{o\}$
find $\{\mu_r\}_{r=1}^{L}\subseteq\mathbb{R}_+$ and $\{S_r\}_{r=1}^{L}\subseteq 2^V$ s.t. $||f(a)-f(b)||_1=\sum_{r=1}^{L}\mu_r\delta_{S_r}(a,b)$, $\forall a,b\in V\cup\{o\}$
for $r=1$ *to* L **do**
    if $o\in S_r$ **then**
        swap S_r with $V\cup\{o\}\setminus S_r$
output $\text{argmin}_{r=1,\ldots,L}\{(\delta_{G_F}(S_r)+\sum_{a\in S_r}y_a)/(\sum_{i=1}^{g}\text{Cross}(S_r,i)z_i)\}$

Let us now analyze Algorithm 3. Note that:

$$\min_{r=1,\ldots,L}\left\{\frac{\delta_{G_F}(S_r)+\sum_{a\in S_r}y_a}{\sum_{i=1}^{g}\text{Cross}(S_r,i)z_i}\right\}\leq\frac{\sum_{r=1}^{L}\mu_r\left(\sum_{e=(a,b)\in\Gamma(S_r)}c_e\delta_{S_r}(a,b)+\sum_{a\in S_r}y_a\right)}{\sum_{r=1}^{L}\mu_r\sum_{i=1}^{g}\text{Cross}(S_r,i)z_i}.$$

Focusing on the numerator we get that:

$$\begin{aligned}
&\sum_{r=1}^{L}\mu_r\left(\sum_{e=(a,b)\in\Gamma(S_r)}c_e\delta_{S_r}(a,b)+\sum_{a\in S_r}y_a\right)\\
&=\sum_{e=(a,b)\in E\setminus F}c_e\sum_{r=1}^{L}\mu_r\delta_{S_r}(a,b)+\sum_{a\in V}y_a\sum_{r=1}^{L}\mu_r\delta_{S_r}(o,a)\\
&=\sum_{e=(a,b)\in E\setminus F}c_e||f(a)-f(b)||_1+\sum_{a\in V}y_a||f(a)-f(o)||_1\\
&\leq D\left(\sum_{e=(a,b)\in E\setminus F}c_e||\mathbf{v}_a-\mathbf{v}_b||_2^2+\sum_{a\in V}y_a||\mathbf{v}_a||_2^2\right).
\end{aligned}$$

The first equality follows from changing the order of summation and the fact that $o\notin S_r$, $\forall r=1,\ldots,L$, i.e., $\mathbf{1}_{\{a\in S_r\}}=\delta_{S_r}(o,a)$. The inequality follows from Theorem 2 and Constraint (6).

Focusing on the denominator and choosing an arbitrary spanning tree $T_{i,j} \in \mathcal{T}_{i,j}$, we get that:

$$\begin{aligned}
\sum_{r=1}^{L} \mu_r \sum_{i=1}^{g} \text{Cross}(S_r, i) z_i &= \sum_{r=1}^{L} \mu_r \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} \mathbf{1}_{\{S_r \in C(Y_{i,j})\}} \\
&\geq \sum_{r=1}^{L} \mu_r \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} \frac{1}{|Y_{i,j}| - 1} \sum_{(a,b) \in T_{i,j}} \delta_{S_r}(a,b) \\
&\geq \frac{1}{X-1} \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} \sum_{(a,b) \in T_{i,j}} \sum_{r=1}^{L} \mu_r \delta_{S_r}(a,b) \\
&= \frac{1}{X-1} \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} \sum_{(a,b) \in T_{i,j}} ||f(a) - f(b)||_1 \\
&\geq \frac{1}{X-1} \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} \sum_{(a,b) \in T_{i,j}} ||\mathbf{v}_a - \mathbf{v}_b||_2^2 \\
&\geq \frac{1}{X-1} \sum_{i=1}^{g} z_i \sum_{j=1}^{\ell_i} s_{i,j} \geq \frac{1}{X-1}.
\end{aligned}$$

The first equality follows from the definition of $\text{Cross}(S_r, i)$. The first inequality follows from the fact that any spanning tree in $\mathcal{T}_{i,j}$ can cross the cut $S_r \in C(Y_{i,j})$ at most $|Y_{i,j}|-1$ times. The second inequality follows from changing the order of summation and from $|Y_{i,j}| \leq X$. The third inequality follows from Theorem 2 and the fact that if $(a,b) \in T_{i,j}$ then $a, b \in Y_{ß,j} \subseteq U$. The before last inequality follows from Constraint (7), whereas the last inequality follows from Constraint (8). The second step of the proof is concluded by combining the upper bound on the nominator with the lower bound on the denominator, along with the first step of the proof. ◀

5.2 Proving Theorem 4

As previously mentioned we follow the footsteps of Young's iterative method [23]. Please refer to Appendix A for the details.

References

1. Sanjeev Arora, James R. Lee, and Assaf Naor. Fréchet embeddings of negative type metrics. *Discret. Comput. Geom.*, 38(4):726–739, 2007.
2. Sanjeev Arora, James R. Lee, and Assaf Naor. Euclidean distortion and the sparsest cut. *J. Amer. Math. Soc.*, 21(1):1–21, 2008.
3. Adi Avidor and Michael Langberg. The multi-multiway cut problem. *Theor. Comput. Sci.*, 377(1-3):35–42, 2007.
4. Niv Buchbinder, Joseph (Seffi) Naor, and Roy Schwartz. Simplex partitioning via exponential clocks and the multiway-cut problem. *SIAM J. Comput.*, 47(4):1463–1482, 2018.
5. Niv Buchbinder, Roy Schwartz, and Baruch Weizman. Simplex transformations and the multiway cut problem. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017*, pages 2400–2410. SIAM, 2017.
6. Gruia Călinescu, Howard J. Karloff, and Yuval Rabani. An improved approximation algorithm for MULTIWAY CUT. *J. Comput. Syst. Sci.*, 60(3):564–574, 2000.

7 Shuchi Chawla, Anupam Gupta, and Harald Räcke. Embeddings of negative-type metrics and an improved approximation to generalized sparsest cut. *ACM Trans. Algorithms*, 4(2):22:1–22:18, 2008.

8 Chandra Chekuri, Sudipto Guha, and Joseph (Seffi) Naor. The steiner k-cut problem. *SIAM J. Discrete Math.*, 20(1):261–271, 2006.

9 Elias Dahlhaus, David S. Johnson, Christos H. Papadimitriou, Paul D. Seymour, and Mihalis Yannakakis. The complexity of multiterminal cuts. *SIAM J. Comput.*, 23(4):864–894, 1994.

10 Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15 of *Algorithms and combinatorics*. Springer, 1997.

11 Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J. Comput.*, 25(2):235–251, 1996.

12 Anupam Gupta, Viswanath Nagarajan, and R. Ravi. An improved approximation algorithm for requirement cut. *Operations Research Letters*, 38(4):322–325, 2010.

13 David R. Karger, Philip N. Klein, Clifford Stein, Mikkel Thorup, and Neal E. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. *Math. Oper. Res.*, 29(3):436–461, 2004.

14 Philip N. Klein, Serge A. Plotkin, Satish Rao, and Éva Tardos. Approximation algorithms for steiner and directed multicuts. *J. Algorithms*, 22(2):241–269, 1997.

15 Philip N. Klein, Satish Rao, Ajit Agrawal, and R. Ravi. An approximate max-flow min-cut relation for unidirected multicommodity flow, with applications. *Combinatorica*, 15(2):187–202, 1995.

16 Frank Thomson Leighton and Satish Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *J. ACM*, 46(6):787–832, 1999.

17 Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

18 Viswanath Nagarajan and R. Ravi. Approximation algorithms for requirement cut on graphs. *Algorithmica*, 56(2):198–213, 2010.

19 Joseph (Seffi) Naor and Yuval Rabani. Tree packing and approximating k-cuts. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA*, pages 26–27, 2001.

20 R. Ravi and Amitabh Sinha. Approximating k-cuts via network strength. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA*, pages 621–622, 2002.

21 Huzur Saran and Vijay V. Vazirani. Finding k cuts within twice the optimal. *SIAM J. Comput.*, 24(1):101–108, 1995.

22 Ankit Sharma and Jan Vondrák. Multiway cut, pairwise realizable distributions, and descending thresholds. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014*, pages 724–733, 2014.

23 Neal E. Young. Sequential and parallel algorithms for mixed packing and covering. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pages 538–546, 2001.

A Proving Theorem 4

For simplicity of presentation, for the remainder of this section, we assume that $F \subseteq E$ is given and fixed. Let M be a guess of the value of OPT, and given M define the following polytope capturing (LP_F):

$$\mathcal{Q}(M) = \Big\{\lambda \in \mathbb{R}^{2^V} : \frac{1}{M}\sum_{S\subseteq V}\frac{1}{2}\delta_{G_F}(S)\lambda_S \leq 1 \qquad , \tag{9}$$

$$\sum_{S:u\in S}\lambda_S \leq 1 \qquad \forall u \in V, \tag{10}$$

$$\frac{1}{r'_i}\sum_{j=1}^{\ell_i}\sum_{S\in C(Y_{i,j})}\lambda_S \geq 1 \qquad \forall i = 1,\ldots,g, \tag{11}$$

$$\lambda_S \geq 0 \qquad \forall S \subseteq V\Big\}.$$

We refer to Constraints (9) and (10) as the packing constraints, and Constraint (11) as the covering constraint. For a sufficiently large parameter N, to be determined later, we use the smoothed definitions of the max and min functions for the packing and covering constraints of $\mathcal{Q}(M)$, respectively:

$$\ell max(\lambda) \triangleq \frac{1}{N}\ln\left(R(\lambda) + \sum_{u\in V} y(\lambda,u)\right) \quad \text{and} \quad \ell min(\lambda) \triangleq -\frac{1}{N}\ln\left(\sum_{i=1}^{g} z(\lambda,i)\right),$$

where:

$$R(\lambda) \triangleq \exp\left(\frac{N}{M}\sum_{S\subseteq V}\frac{1}{2}\delta_{G_F}(S)\lambda_S\right)$$

$$y(\lambda,u) \triangleq \exp\left(N\sum_{S:u\in S}\lambda_S\right)$$

$$z(\lambda,i) \triangleq \exp\left(-\frac{N}{r'_i}\sum_{j=1}^{\ell_i}\sum_{S\in C(Y_{i,j})}\lambda_S\right).$$

To simplify presentation, we denote the following value of a cut $S \subseteq V$ by $\phi_\lambda(S)$:

$$\phi_\lambda(S) \triangleq \frac{\sum_{i=1}^{g} z(\lambda,i)}{R(\lambda)+\sum_{u\in V} y(\lambda,u)} \cdot \frac{\frac{1}{M}\frac{1}{2}\delta_{G_F}(S)R(\lambda) + \sum_{u\in S} y(\lambda,u)}{\sum_{i=1}^{g}\frac{1}{r'_i}\mathrm{Cross}(S,i)z(\lambda,i)}.$$

There are two important things to note regarding ϕ_λ. First, for every $S \subseteq V$:

$$\phi_\lambda(S) = \frac{\frac{\partial \ell max(\lambda)}{\partial \lambda_S}}{\frac{\partial \ell min(\lambda)}{\partial \lambda_S}}. \tag{12}$$

Observation (12) plays a crucial role when analyzing Young's iterative method applied to our setting. Second, $\phi_\lambda(S)$ equals the value of S when considering the following instance $(G, F, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c_\lambda, y_\lambda, z_\lambda)$ of (SRC):

$$c_\lambda(e) = \frac{\frac{1}{2M}R(\lambda)c_e}{R(\lambda)+\sum_{u\in V} y(\lambda,u)} \qquad \forall e \in E\setminus F$$

$$y_\lambda(u) = \frac{y(\lambda,u)}{R(\lambda)+\sum_{u\in V} y(\lambda,u)} \qquad \forall u \in V$$

$$z_\lambda(i) = \frac{\frac{1}{r'_i}z(\lambda,i)}{\sum_{i=1}^{g} z(\lambda,i)} \qquad \forall i = 1,\ldots,g.$$

The following lemma proves that there exists a solution to the above instance of (SRC) whose value is at most 1, assuming our guess for M is not smaller than OPT.

▶ **Lemma 9.** *For every $\lambda \in \mathbb{R}^{2^V}$, if $M \geq OPT$ then there exists a cut $S \subseteq V$ such that $\phi_\lambda(S) \leq 1$.*

Proof. We define two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^{2^V}$, indexed by S, as follows:

$$a_S \triangleq \frac{\frac{1}{M}\frac{1}{2}\delta_{G_F}(S)R(\lambda) + \sum_{u\in S} y(\lambda,u)}{R(\lambda) + \sum_{u\in V} y(\lambda,u)} \quad \text{and} \quad b_S \triangleq \frac{\sum_{i=1}^{g} \frac{1}{r'_i}\text{Cross}(S,i)z(\lambda,i)}{\sum_{i=1}^{g} z(\lambda,i)},$$

and prove that $\langle \mathbf{a}, \lambda^* \rangle \leq 1$ and $\langle \mathbf{b}, \lambda^* \rangle \geq 1$, for some non-negative vector $\lambda^* \in \mathbb{R}_+^{2^V}$. Since $\mathbf{a}$, $\mathbf{b}$, and λ^* are all non-negative, we can conclude that there exists $S \subseteq V$ such that $a_S \leq b_S$. This will conclude the proof.

We note that since $M \geq \text{OPT}$, $\mathcal{Q}(M)$ is non-empty. The reason for the latter is that $\lambda^* \in \mathcal{Q}(M)$, where λ^* is an optimal solution to the (RC) problem applied to G_F (as defined in the proof of Lemma 3). Hence, we know that:

$$\sum_{S\subseteq V} \frac{1}{2}\delta_{G_F}(S)\lambda^*_S \leq \text{OPT} \tag{13}$$

$$\sum_{S:u\in S} \lambda^*_S \leq 1 \qquad \forall u \in V \tag{14}$$

$$\sum_{j=1}^{\ell_i} \sum_{S\in C(Y_{i,j})} \lambda^*_S \geq r'_i \qquad \forall i = 1,\ldots,g. \tag{15}$$

We note that (13) above follows from Lemma 3. First, let us prove that $\langle \mathbf{a}, \lambda^* \rangle \leq 1$:

$$\begin{aligned}
\langle \mathbf{a}, \lambda^* \rangle &= \sum_{S\subseteq V} \frac{\frac{1}{M}\frac{1}{2}\delta_{G_F}(S)R(\lambda) + \sum_{u\in S} y(\lambda,u)}{R(\lambda) + \sum_{u\in V} y(\lambda,u)}\lambda^*_S \\
&= \frac{\frac{R(\lambda)}{M}\sum_{S\subseteq V}\frac{1}{2}\delta_{G_F}(S)\lambda^*_S + \sum_{u\in V} y(\lambda,u)\sum_{S:u\in S}\lambda^*_S}{R(\lambda) + \sum_{u\in V} y(\lambda,u)} \\
&\leq \frac{\frac{R(\lambda)}{M}\text{OPT} + \sum_{u\in V} y(\lambda,u)}{R(\lambda) + \sum_{u\in V} y(\lambda,u)} \leq 1.
\end{aligned}$$

The second equality follows from changing the order of summation. The first inequality follows from (13) and (14), whereas the last inequality follows since $M \geq \text{OPT}$. Second, let us prove that $\langle \mathbf{b}, \lambda^* \rangle \geq 1$:

$$\begin{aligned}
\langle \mathbf{b}, \lambda^* \rangle &= \sum_{S\subseteq V} \frac{\sum_{i=1}^{g}\frac{1}{r'_i}\text{Cross}(S,i)z(\lambda,i)}{\sum_{i=1}^{g} z(\lambda,i)}\lambda^*_S \\
&= \frac{\sum_{i=1}^{g}\frac{1}{r'_i} z(\lambda,i)\sum_{j=1}^{\ell_i}\sum_{S\in C(Y_{i,j})}\lambda^*_S}{\sum_{i=1}^{g} z(\lambda,i)} \geq 1.
\end{aligned}$$

The second equality follows from changing the order of summation, whereas the inequality follows from (15). ◀

The following lemma establishes the connection between the above and our algorithm for (approximately) solving (SRC), i.e., Theorem 8.

▶ **Lemma 10.** *Given $\lambda \in \mathbb{R}^{2^V}$, if $M \geq OPT$ then executing the algorithm of Theorem 8 on the instance $(G, F, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c_\lambda, y_\lambda, z_\lambda)$ of (SRC) yields a cut $\tilde{S} \subseteq V$ satisfying: $\phi_\lambda(\tilde{S}) \leq (X-1)D$.*

Proof. Let S^* be an optimal solution to the instance $(G, F, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c_\lambda, y_\lambda, z_\lambda)$ of (SRC), or equivalently, a cut that minimizes $\phi_\lambda(S^*)$. Since $M \geq \mathrm{OPT}$, we can apply Lemma 9 and obtain that $\mathrm{OPT}_{\mathrm{SRC}} \leq 1$ for the given instance. Theorem 8 concludes the proof. ◀

We are now ready to present Young's iterative approach adapted to (LP_F), Algorithm 4. The parameter $\zeta > 0$ in the input determines the step size of the algorithm and will be determined later.

Algorithm 4 Solving (LP_F).

Input: $G = (V, E), F \subseteq E, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c : E \to \mathbb{R}_+, \zeta > 0$
Output: $\lambda \in \mathbb{R}_+^{2^V}$
$\lambda \leftarrow \mathbf{0}$
while $\exists i = 1, \ldots, g$ *s.t.* $\frac{1}{r'_i} \sum_{j=1}^{\ell_i} \sum_{S \in C(Y_{i,j})} \lambda_S < 1$ **do**
  apply Theorem 8 on instance $(G, F, \{\{Y_{i,j}\}_{j=1}^{\ell_i}\}_{i=1}^{g}, c_\lambda, y_\lambda, z_\lambda)$ of (SRC) to obtain $\tilde{S}$
  $\lambda_{\tilde{S}} \leftarrow \lambda_{\tilde{S}} + \zeta$
output λ

The following lemma is used to upper bound the step size ζ in Algorithm 4 and it follows directly by adapting Lemma 1 of [23] to our setting.

▶ **Lemma 11.** *For every $0 < \varepsilon \leq 1$ and $\zeta > 0$ such that $\zeta \leq \varepsilon \min\{1, 1/X, (2M)/(\sum_{e \in E \setminus F} c_e)\}$, the following two hold for every $S \subseteq V$:*

$$\frac{\ell max(\lambda + \zeta \mathbf{1}_S) - \ell max(\lambda)}{\zeta(1+\varepsilon)} \leq \frac{\partial \ell max(\lambda)}{\partial \lambda_S} = \frac{\frac{1}{M} \cdot \frac{1}{2}\delta_{G_F}(S) \cdot R(\lambda) + \sum_{u \in S} y(\lambda, u)}{R(\lambda) + \sum_{u \in V} y(\lambda, u)} \tag{16}$$

$$\frac{\ell min(\lambda + \zeta \mathbf{1}_S) - \ell min(\lambda)}{\zeta(1-\varepsilon/2)} \geq \frac{\partial \ell min(\lambda)}{\partial \lambda_S} = \frac{\sum_{i=1}^{g} \frac{1}{r'_i} \cdot Cross(S, i) z(\lambda, i)}{\sum_{i=1}^{g} z(\lambda, i)}. \tag{17}$$

The following lemma states that when Algorithm 4 terminates, assuming $M \geq \mathrm{OPT}$, it produces an approximate solution to $\mathcal{Q}(M)$.

▶ **Lemma 12.** *For every $0 < \varepsilon \leq 1$, $\zeta > 0$ satisfying the conditions of Lemma 11, and $N = \varepsilon^{-1} \ln(g(n+1))$, when Algorithm 4 terminates, assuming $M \geq OPT$, it outputs $\lambda \in \mathbb{R}_+^{2^V}$ satisfying:*

$$\sum_{S: u \in S} \lambda_S \leq \alpha \qquad \forall u \in V \tag{18}$$

$$\frac{1}{r'_i} \sum_{j=1}^{\ell_i} \sum_{S \in C(Y_{i,j})} \lambda_S \geq \beta \qquad \forall i = 1, \ldots, g \tag{19}$$

$$\frac{1}{M} \sum_{S \subseteq V} \frac{1}{2} \delta_{G_F}(S) \lambda_S \leq \gamma. \tag{20}$$

In the above $\alpha = \gamma = \varepsilon + \frac{(1+\varepsilon)(1+2\varepsilon)}{1-\varepsilon/2}(X-1)D$ and $\beta = 1$.

Proof. From the stopping condition of Algorithm 4, it is clear that (19) holds with $\beta = 1$, i.e., the covering constraints are satisfied. Let us now focus on the remaining two covering constraints. Denote the sequence of cuts Algorithm 4 increased by: $S_1, \ldots, S_L$ (note that some cuts might appear several times in the sequence), and by λ the output, i.e., $\lambda = \zeta \sum_{r=1}^{L} \mathbf{1}_{S_r}$. For simplicity of presentation, we denote by λ^r the solution Algorithm 4 maintains after the r^{th} iteration, i.e., $\lambda^r = \zeta \sum_{s=1}^{r} \mathbf{1}_{S_s}$. Note that $\lambda^0 = \mathbf{0}$ and $\lambda^L = \lambda$. Let us now upper bound the worst packing constraint upon termination of the algorithm:

$$\begin{aligned}
&\max\left\{\frac{1}{M}\sum_{S\subseteq V}\frac{1}{2}\delta_{G_F}(S)\lambda_S, \max_{u\in V}\left\{\sum_{S:u\in S}\lambda_S\right\}\right\} \leq \ell max(\lambda)\\
&= \frac{\ln(n+1)}{N} + \sum_{r=1}^{L}\left(\ell max\left(\lambda^r\right) - \ell max\left(\lambda^{r-1}\right)\right)\\
&\leq \frac{\ln(n+1)}{N} + \frac{1+\varepsilon}{1-\varepsilon/2}\sum_{r=1}^{L}\phi_{\lambda^{r-1}}(S_r)\left(\ell min\left(\lambda^r\right) - \ell min\left(\lambda^{r-1}\right)\right)\\
&\leq \frac{\ln(n+1)}{N} + \frac{1+\varepsilon}{1-\varepsilon/2}(X-1)D\sum_{r=1}^{L}\left(\ell min\left(\lambda^r\right) - \ell min\left(\lambda^{r-1}\right)\right)\\
&= \frac{\ln(n+1)}{N} + \frac{1+\varepsilon}{1-\varepsilon/2}(X-1)D\frac{\ln g}{N} + \frac{1+\varepsilon}{1-\varepsilon/2}(X-1)D\ell min(\lambda)\\
&\leq \varepsilon + \frac{\varepsilon(1+\varepsilon)}{1-\varepsilon/2}(X-1)D + \frac{1+\varepsilon}{1-\varepsilon/2}(X-1)D\min_{i=1,\ldots,g}\left\{\frac{1}{r'_i}\sum_{j=1}^{\ell_i}\sum_{S\in C(Y_{i,j})}\lambda_S\right\}\\
&\leq \varepsilon + \frac{\varepsilon(1+\varepsilon)}{1-\varepsilon/2}(X-1)D + \frac{1+\varepsilon}{1-\varepsilon/2}(X-1)D\cdot(1+\zeta X)\\
&\leq \varepsilon + \frac{\varepsilon(1+\varepsilon)}{1-\varepsilon/2}(X-1)D + \frac{1+\varepsilon}{1-\varepsilon/2}(X-1)D\cdot(1+\varepsilon)\\
&= \varepsilon + \frac{(1+\varepsilon)(1+2\varepsilon)}{1-\varepsilon/2}(X-1)D.
\end{aligned}$$

The first inequality follows from the definition of ℓmax. Since $\ell max(\lambda^0) = \ell max(\mathbf{0}) = \ln(n+1)/N$, the first equality follows. The second inequality follows from Lemma 11 and the definition of ϕ. We note that the third inequality follows from Lemma 10 and how S_r is chosen by Algorithm 4. Since $\ell min(\lambda^0) = \ell min(\mathbf{0}) = -\ln g/N$, the second equality follows. The fourth inequality follows from the choice of N and the definition of ℓmin. Note that the fifth inequality follows from the stopping condition of Algorithm 4, i.e., the algorithm stops once all coverings constraints are satisfied. The last inequality follows from the restrictions on ζ. ◀

Proof of Theorem 4. All that remains is to choose ζ, such that for every guess of M Algorithm 4 performs a polynomial number of iterations. We do this by tracking the following potential: $\Phi(\lambda) \triangleq \sum_{u\in V}\sum_{S:u\in S}\lambda_S$. Note that initially $\Phi(\lambda) = \Phi(\mathbf{0}) = 0$. From guarantee (18) of Lemma 12, we know that once Algorithm 4 terminates the value of $\Phi(\lambda)$ cannot exceed $n(\varepsilon + (1+\varepsilon)(1+2\varepsilon)/(1-\varepsilon/2)\cdot(X-1)D)$. Setting the step size $\zeta = \varepsilon\min\{1, 1/X, (2M)/(\sum_{e\in E\setminus F} c_e)\}$ and choosing $\varepsilon = 1$ implies that Algorithm 4 performs at most $O(nXD/\zeta)$ iterations. Applying standard weight rescaling techniques, we can assume without loss of generality, that $1 \leq c_e \leq \text{poly(n)}$, $\forall e \in E$. Thus, given a guess M, Algorithm 4 performs at most a polynomial number of iterations (recall that $M \geq 1$ due to the rescaling). Since all edge weights are rescaled as above, one can find in polynomial time a value M such that $\text{OPT} \leq M \leq 2\text{OPT}$. This concludes the proof. ◀

Parametrized Metrical Task Systems

Sébastien Bubeck
Microsoft Research, Redmond, WA, USA
sebubeck@microsoft.com

Yuval Rabani
Hebrew University of Jerusalem, Israel
yrabani@cs.huji.ac.il

Abstract

We consider parametrized versions of metrical task systems and metrical service systems, two fundamental models of online computing, where the constrained parameter is the number of possible distinct requests m. Such parametrization occurs naturally in a wide range of applications. Striking examples are certain power management problems, which are modeled as metrical task systems with $m = 2$. We characterize the competitive ratio in terms of the parameter m for both deterministic and randomized algorithms on hierarchically separated trees. Our findings uncover a rich and unexpected picture that differs substantially from what is known or conjectured about the unparametrized versions of these problems. For metrical task systems, we show that deterministic algorithms do not exhibit any asymptotic gain beyond one-level trees (namely, uniform metric spaces), whereas randomized algorithms do not exhibit any asymptotic gain even for one-level trees. In contrast, the special case of metrical service systems (subset chasing) behaves very differently. Both deterministic and randomized algorithms exhibit gain, for m sufficiently small compared to n, for any number of levels. Most significantly, they exhibit a large gain for uniform metric spaces and a smaller gain for two-level trees. Moreover, it turns out that in these cases (as well as in the case of metrical task systems for uniform metric spaces with m being an absolute constant), deterministic algorithms are essentially as powerful as randomized algorithms. This is surprising and runs counter to the ubiquitous intuition/conjecture that, for most problems that can be modeled as metrical task systems, the randomized competitive ratio is polylogarithmic in the deterministic competitive ratio.

2012 ACM Subject Classification Theory of computation → Theory and algorithms for application domains

Keywords and phrases online computing, competitive analysis, metrical task systems

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.54

Category APPROX

Funding *Yuval Rabani*: Research supported in part by ISF grants 956-15 and 2553-17. Part of this work was done while visiting Microsoft Research at Redmond.

1 Introduction

1.1 Motivation

Metrical task systems are a general framework proposed in the early days of online computing to model a wide range of problems. In this model, an algorithm controls its state among n possible states. It is presented with a sequence of requests. Each request indicates the cost of serving it in each of the n states. The algorithm must choose a state to serve the request, then pay both the transition cost and the service cost. It competes against the optimal offline solution, and is c-competitive iff for any request sequence it pays at most c times this yardstick (plus an allowed additive term independent of the request sequence). The transition costs are assumed to be distances in a finite metric space whose points are the states. The seminal paper [12] introduces this model and also solves completely a fundamental


© Sébastien Bubeck and Yuval Rabani;
licensed under Creative Commons License CC-BY



problem regarding the proposed model. It gives a tight $2n-1$ upper and lower bound on the competitive ratio of any deterministic algorithm. The paper also raises the same question regarding randomized algorithms. For a uniform metric space, it gives a lower bound of $\mathcal{H}_n$ and a nearly tight upper bound of $2\mathcal{H}_n$, where $\mathcal{H}_n = 1 + \frac{1}{2} + \cdots + \frac{1}{n}$ is the n-th harmonic number. In light of this result, a natural folklore conjecture asserts that in any metric space the randomized competitive ratio is $\Theta(\log n)$. This conjecture turned out to be far more elusive than the deterministic case. Currently, the best universal bounds are a lower bound of $\Omega(\log n/\log\log n)$ [7, 9] and a very recent upper bound of $O(\log^2 n)$ [13].

Many online problems can be modeled as special cases of metrical task systems. For instance, paging [31] and more generally the k-server problem [27] are task systems on the state space of k-tuples of points in an underlying metric space. But the best bounds that are known for these problems are much lower than the general bounds that apply to any metrical task system. Clearly, there are constraints on the allowed requests. For instance, in the k-server problem in an n-point metric space, the state space has size $\binom{n}{k}$, but the number of possible requests is just n, and they have only 0 or ∞ values. Such constraints often characterize concrete applications of metrical task systems (e.g., various problems in networks, see [8] and the references therein). A striking example of the gain from instances with a restricted set of requests is the power management problem [25]. It allows only two possible requests, and admits an $O(1)$-competitive algorithm [2]. In view of these examples, it is natural to seek a meaningful measure of instance complexity that will provide at least partial prediction of the better bounds in special cases, a prediction that may be applied to new problems. An obvious choice is the cardinality m of the set of possible requests. (Clearly, this question is interesting only for $m \ll n$, otherwise the known general lower bounds kick in.) Examining request sequences drawn from a given set is proposed explicitly in [15]. They give a deterministic algorithm that in uniform metric spaces achieves a competitive ratio of $O(\log n)$ times the (unknown) best possible bound.

Motivated by the k-server example, the restriction of the requests to have costs in $\{0, \infty\}$ is a natural and attractive problem of chasing subsets of points in a finite metric space. Indeed, this problem setting was proposed in [17], dubbed metrical service systems. We note that this is part of a broader theme of online chasing and searching, and other seminal papers in this theme include [28, 3, 23]. The focus of the metrical service systems literature (e.g. [17, 20, 29, 14, 19, 16]) is the *width* of the requests, which is the maximum size of a requested set. In view of the above discussion, it is equally natural to ask about chasing sets drawn from a small-sized pool of possible requests and deriving bounds that are independent of the width.

1.2 Our results

Our main contribution is an explicit analysis of the competitive ratio of metrical task systems and metrical service systems with m possible requests, $m \ll n$. Perhaps surprisingly, we show that unlike the general case, the answer here depends on the metric space. In particular, for metrical task systems we show that in uniform metric spaces, restricting thus the request sequence improves dramatically the deterministic competitive ratio, which we characterize tightly, up to constant factors. In contrast, there exist other metric spaces, namely two-level HSTs (uniform metrics are one-level HSTs), that exhibit no deterministic asymptotic improvement, even if $m = 2$. Perhaps even more surprisingly, in uniform metric spaces the randomized competitive ratio does not improve asymptotically even if $m = 2$, and in this case the deterministic competitive ratio is asymptotically the same as the randomized competitive ratio.

We also analyze unrestricted width metrical service systems with m possible requests in uniform metric spaces versus two-level HSTs. This case, too, exhibits complex and fascinating behavior, and furthermore it is not identical to the general case. In sufficiently large uniform metric spaces the restriction to m requests improves both the deterministic and the randomized competitive ratio. However the two ratios are asymptotically the same, linear in m and independent of the size of the metric. The two-level HSTs case also deviates from the general case. As with uniform metric spaces, both the deterministic and the randomized competitive ratios improve and are independent of the size of the metric space. However, both the deterministic and the randomized competitive ratios are now exponential in m (the base of the exponent may be different in the two cases). We discuss the surprising aspects of these results and their features that persist beyond two levels in the following sections.

We note that our results, most strikingly the bounds for $m = 2$ metrical task systems in uniform metric spaces, refute a folklore conjecture that for most problems that can be cast as metrical task systems, the randomized competitive ratio is polylogarithmic in the deterministic competitive ratio.

Finally, we relate some of our results to the question of quantifying the amount of information an adversary needs in order to force an online algorithm to perform poorly.

2 Definitions and Results

A *metrical task system* instance consists of a finite metric space $\mathcal{M} = (X, d)$, an initial state $s_0 \in X$, and a sequence of requests $\rho_1, \rho_2, \ldots, \rho_L$, where for every $t \in \{1, 2, \ldots, L\}$, the request ρ_t is a cost function that maps X to $\mathbb{R}_+ \cup \{+\infty\}$. We will denote $n \triangleq |X|$. A solution consists of a choice of states $s_1, s_2, \ldots, s_L \in X$, which incurs a total cost of

$$\sum_{t=1}^{L} \left(d(s_{t-1}, s_t) + \rho_t(s_t) \right).$$

A deterministic online algorithm chooses, for every $t = 1, 2, \ldots, L$, the next state s_t based only on $\mathcal{M}$ and $\rho_1, \rho_2, \ldots, \rho_t$, without knowing the suffix of the requests sequence or even its length L. A randomized online algorithm is a probability distribution over deterministic online algorithms, and its cost is defined to be the expectation of the cost of the deterministic algorithms in the support of the distribution.

We use the following notation. For a sequence of requests ρ, let $z^*(\rho)$ denote the optimal cost of serving ρ. Define $c_{\mathcal{M}}^{\det}$ to be the infimum over c such that there exists a constant a and a deterministic online algorithm A such that for every sequence of requests ρ, the cost of A of serving ρ is at most $c \cdot z^*(\rho) + a$. Define similarly $c_{\mathcal{M}}^{\mathrm{rand}}$ for randomized algorithms. The following theorem is well-known:

▶ **Theorem 1** (Borodin et al. [12]). *For every finite metric space $\mathcal{M}$,*

$$c_{\mathcal{M}}^{\det} = 2n - 1.$$

It is also conjectured that $c_{\mathcal{M}}^{\mathrm{rand}} = \Theta(\log n)$. The conjecture has been established in special cases. For instance tight bounds are known in uniform metric spaces. The currently best known bounds (following substantial previous work [26, 11, 15, 10, 6, 24, 30, 22, 4, 1]) that hold for every metric space are (see below for the definition of an HST):

▶ **Theorem 2** (Bartal et al. [7, 9]). *For every finite metric space $\mathcal{M}$,*

$$c_{\mathcal{M}}^{\mathrm{rand}} = \Omega\left(\frac{\log n}{\log \log n} \right).$$

▶ **Theorem 3** (Bubeck et al. [13]). *For every HST $\mathcal{M}$,*

$$c_{\mathcal{M}}^{\mathrm{rand}} = O(\log n),$$

and therefore, for every finite metric space $\mathcal{M}$,

$$c_{\mathcal{M}}^{\mathrm{rand}} = O(\log^2 n).$$

We are interested in analyzing metrical task systems with constrained requests. The primary constraint that we investigate in this paper is the following: there exists a finite set $\mathcal{R} \subsetneq (\mathbb{R}_+ \cup \{+\infty\})^X$ such that all the requests lie in $\mathcal{R}$. We will denote $m \triangleq |\mathcal{R}|$. Let $c_{\mathcal{M},\mathcal{R}}^{\mathrm{det}}$ ($c_{\mathcal{M},\mathcal{R}}^{\mathrm{rand}}$, respectively) denote the deterministic (randomized, respectively) competitive ratio when requests are restricted to the set $\mathcal{R}$. Also let $c_{\mathcal{M},m}^{\mathrm{det}}$ ($c_{\mathcal{M},m}^{\mathrm{rand}}$, respectively) denote the supremum of $c_{\mathcal{M},\mathcal{R}}^{\mathrm{det}}$ ($c_{\mathcal{M},\mathcal{R}}^{\mathrm{rand}}$, respectively) over $|\mathcal{R}| = m$. We find that unlike the general case, the competitive ratio that can be guaranteed for constrained metrical task systems depends crucially on the underlying metric space $\mathcal{M}$. In particular, we study a class of ultrametrics called hierarchically separated trees (HSTs), invented in [5]. HSTs play a critical role in both the best known upper bounds and the best known lower bounds for metrical task systems, as well as other problems involving metric spaces. For metrical task systems, the best known upper bounds rely on an asymptotically optimal approximation of any metric space by a convex combination of HSTs, discovered in [18]. The best known lower bounds rely on a lower bound on HSTs and the fact that any metric space contains a large approximate HST subspace [7, 9]. More concretely, an L-level HST is a metric space defined on the leaves of a leveled rooted node-weighed tree with $L+1$ levels. The leaves are at level 0, the root is the unique node at level L, each node other than the root has a parent at one level above its own, all the nodes at level i have the same weight, and this weight increases rapidly with the level number. The distance between two leaves is the weight of their least common ancestor. (We note that HSTs are often defined alternatively without the uniformity constraint on levels.) This defines an ultrametric (which is in particular a metric) on the leaves.

We exhibit on the one hand that for uniform metric spaces (which are in the above terms one-level HSTs), restricting m helps immensely deterministic algorithms, but not randomized algorithms:

▶ **Theorem 4.** *If $\mathcal{M}$ is a uniform metric space, then for every $m \leq n$,*

$$c_{\mathcal{M},m}^{\mathrm{det}} = \Theta(m \log(en/m)),$$

and

$$c_{\mathcal{M},2}^{\mathrm{rand}} = \Theta(c_{\mathcal{M}}^{\mathrm{rand}}) = \Theta(\log n).$$

On the other hand, we study what is in some sense just a slightly more sophisticated family of metric spaces. We consider the following family of two-level HSTs, which we call *paired-uniform*. Let n be an even number of points. Partition the points into pairs $\{i, i'\}$. (So, according to this notation $i'' = i$.) Let $C \gg 1$ be a large constant to be determined later. Set $d(i, i') = 1$ for all pairs, and set $d(i, j) = C$ for every two points $i \neq j \neq i'$. It turns out that here restricting m helps neither the asymptotic performance of deterministic algorithms, nor that of randomized algorithms:

▶ **Theorem 5.** *If $\mathcal{M}$ is a paired-uniform metric space, then*

$$c_{\mathcal{M},2}^{\mathrm{det}} = \Theta(n),$$

whereas

$$c_{\mathcal{M},2}^{\mathrm{rand}} = \Theta(\log n).$$

Notice that by prior results, the upper bounds hold even for arbitrary requests in arbitrary HSTs.

The lower bound constructions in the proofs of Theorems 4 and 5 use sets of requests that assign many widely varying cost values. Thus, it makes sense to consider requests using costs from a small set of scales, and in particular to consider metrical service systems that use just two scales, 0 or ∞ (a.k.a. set chasing). Let $\hat{c}^{\det}_{\mathcal{M},m}$, ($\hat{c}^{\rm rand}_{\mathcal{M},m}$, respectively) denote the best deterministic (randomized, respectively) competitive ratio that can be achieved for chasing sets drawn from a collection $\mathcal{R}$ of m subsets of points in $\mathcal{M}$. In this case we show that for uniform spaces, restricting m helps *both* deterministic and randomized algorithms. Moreover, if $m = O(\log n)$, the deterministic and the randomized competitive ratios are asymptotically identical. More precisely we have:

▶ **Theorem 6.** *If $\mathcal{M}$ is a uniform metric space, then for every $m \in \mathbb{N}$,*

$$\min\{m, n\} - 1 \leq \hat{c}^{\det}_{\mathcal{M},m} \leq \min\{m, n\},$$

and,

$$\hat{c}^{\rm rand}_{\mathcal{M},m} = \Theta\left(\min\{m, \log n\}\right).$$

Unlike the more general case of Theorem 5, in the case of metrical service systems on two-level HSTs, restricting the number of possible requests helps, but not as much as in uniform metric spaces. In order to state the results, we introduce a bit more notation. Let $\hat{c}^{\det}_{L,m}$ ($\hat{c}^{\rm rand}_{L,m}$, respectively) denote the supremum of $\hat{c}^{\det}_{\mathcal{M},m}$ ($\hat{c}^{\rm rand}_{\mathcal{M},m}$, respectively) over all L-level HSTs $\mathcal{M}$. We prove:

▶ **Theorem 7.** *For every $m \in \mathbb{N}$,*

$$2^{\lfloor m/2 \rfloor - 1} \leq \hat{c}^{\rm rand}_{2,m} \leq \hat{c}^{\det}_{2,m} \leq m2^m.$$

We note that we actually prove a somewhat tighter lower bound of $\binom{m}{\lfloor m/2 \rfloor}$ on $\hat{c}^{\det}_{2,m}$, see Lemma 16. We also note that it is not surprising that for sufficiently large HSTs, the competitive ratio (deterministic or randomized) depends solely on m and the number of levels. This is because one can trim isomorphic nodes from the tree. (Note however that for deterministic algorithms the trivial bound obtained this way is exponentially worse than the ones proposed above.) The surprising aspects of these theorems are (a) that the deterministic competitive ratio nearly matches the randomized competitive ratio, and (b) that for two-level HSTs, a collection of $O(\log \log n)$ set chasing requests is sufficient to generate a randomized lower bound of $\Omega(\log n)$ (while the classical lower bound requires $\Omega(n)$ such requests). In fact, this latter aspect extends to and is amplified in HSTs with any number of levels. So, using just 6 set chasing requests, we can get a lower bound of $\Omega(\log n)$ for infinitely many n, on HSTs with more and more levels. The argument is outlined in Section 4.

Our main results are summarized in Table 1.

Table 1 Summary of our results, for any $n \gg m$.

one-level HSTs (uniform metrics)

	deterministic	randomized
MTS	$\Theta(m \log n)$	$\Theta(\log n)$
MSS	$\Theta(m)$	$\Theta(m)$

two-level HSTs

	deterministic	randomized
MTS	$\Theta(n)$	$\Theta(\log n)$
MSS	$2^{\Theta(m)}$	$2^{\Theta(m)}$

3 Proofs

▶ **Lemma 8.** *Let $\mathcal{M} = (X, d)$ be an n-point uniform metric space. Consider any set of requests $\mathcal{R}$ of cardinality $|\mathcal{R}| = m$ and any set $S \subseteq X$ of cardinality $|S| > m \log(n/m)$. Then, there exists a state $s \in S$ such that for every $r \in \mathcal{R}$,*

$$|\{s' \in S : \ r_{s'} \geq r_s\}| \geq \frac{1}{m} \cdot |S|. \tag{1}$$

Proof. The proof is a simple application of the probabilistic method. Choose $s \in S$ uniformly at random. We have that for every $r \in \mathcal{R}$,

$$\Pr\left[|\{s' \in S : \ r_{s'} \geq r_s\}| < \frac{1}{m} \cdot |S|\right] < \frac{1}{m}.$$

Therefore, applying the union bound,

$$\Pr\left[\exists r \in \mathcal{R} : \ |\{s' \in S : \ r_{s'} \geq r_s\}| < \frac{1}{m} \cdot |S|\right] < 1,$$

so there is a choice of $s \in S$ that satisfies Equation (1). ◀

▶ **Lemma 9.** *Let $\mathcal{M}$ be an n-point uniform metric space. Then, $c^{\det}_{\mathcal{M},m} = O(m \log(en/m))$.*

Proof. Consider the following algorithm. The algorithm works in phases, and these are further partitioned into rounds. In a phase, we will denote by S_r the set S at the end of round r. We define the cost that a state s acrues during some interval I as $\Sigma(I; s) = \sum_{t \in I} \rho_t(s)$. In the beginning of a phase, set $S_0 = X$ and move to a state s that satisfies Equation (1). As soon as at least $|S_{r-1}|/2m$ states in S_{r-1} acrue a cost of at least 1 in the current phase (including the new request), round r ends. When this happens, remove from S_{r-1} the $\lceil |S_{r-1}|/2m \rceil$ states that acrued the highest cost. Set S_r to be the remaining states in S_{r-1}. If $|S_r| \geq m \log(en/m)$, repeat the above process with a new $s \in S_r$. Otherwise, this is the last round. In this last round, use a $(2|S_r| - 1)$-competitive algorithm, restricted to S_r. Execute this algorithm until the optimal cost of serving this round reaches at least 1. Notice that this implies that by the end of the phase, all states in X acrued a cost of at least 1 during the phase. Therefore, the adversary's cost for the entire phase must be at least 1, because it either stayed at one state and paid a service cost of 1, or moved at least once and paid a movement cost of 1.

In order to analyze this algorithm, fix a phase and denote by a_r the total cost that the algorithm acrues in round r of this phase. Also denote by o_r the average over $s \in S_r$ of the minimum between 1 and the cost acrued by s in this phase by the end of round r. (Set $o_0 = 0$.) Notice that $|S_r| \leq \left(1 - \frac{1}{2m}\right) \cdot |S_{r-1}|$. This implies that the number of rounds (including the special last round) is at most $2m \log(n/m) + 1$. Also notice that throughout round r, at each request at least $|S_{r-1}|/m$ states in S_{r-1} pay at least the cost that the algorithm pays, but less than $|S_{r-1}|/2m$ states in S_{r-1} acrued a total cost of at least 1. Thus, the average over S_{r-1} increases by at least $a_r/2m$. Each state in $S_{r-1} \setminus S_r$ contributes $\frac{1}{|S_{r-1}|}$ to this average, and there are $|S_{r-1}|/2m$ such states, so removing them reduces the average by at most $\frac{1}{2m}$. Thus,

$$o_r \geq o_{r-1} + \frac{a_r}{2m} - \frac{1}{2m}.$$

Let $\bar{r}$ be the penultimate round. Notice that for all r, $o_r \leq 1$. Then

$$2m \geq 2m \cdot o_{\bar{r}} \geq \sum_{r=1}^{\bar{r}} a_r - \bar{r} \geq \sum_{r=1}^{\bar{r}} a_r - 2m \log(n/m).$$

Thus, $\sum_{r=1}^{\bar{r}} a_r = O(m \log(en/m))$. The last phase also costs the algorithm $O(m \log(en/m))$. This is because $|S_{\bar{r}}| < m \log(en/m)$, we use a $(2|S_{\bar{r}}| - 1)$-competitive algorithm, and an adversary with the same initial state in this round as the algorithm pays at most 2 – one for moving to the best initial state, and one for serving the request sequence in the round. ◀

▶ **Lemma 10.** *Let $\mathcal{M}$ be an n-point uniform metric space. Then, $c^{\mathrm{rand}}_{\mathcal{M},2} = \Omega(\log n)$.*

Proof. Assume without loss of generality that $n = 2^q$ is a power of 2 (otherwise we can restrict the instance to the power of 2 closest to n; the redundant states can be blocked by assigning to them a cost of ∞ in both elements of $\mathcal{R}$). Let $\{0, 1, \ldots, n-1\}$ be an arbitrary enumeration of the points of $\mathcal{M}$. Let $C = C(n)$ be a large constant. Set $\mathcal{R} = \{r^0, r^1\}$, where $r^0_i = C^{i-n}$, and let $r^1_i = C^{-i-1}$. We use the minimax principle and generate a random request sequence that "beats" every deterministic algorithm. The request sequence is generated in phases. In each phase we choose uniformly at random one state $h \in \{0, 1, \ldots, n-1\}$ where the adversary "hides." Let $h_0 h_1 \cdots h_{q-1}$ be the binary representation of h. This defines a natural sequence of nested intervals in the dyadic partition of the set of states. The interval J_i consists of all the states whose binary representation has the prefix $h_0 \cdots h_i$. For notational convenience, let $J_{-1} = \{0, 1, \ldots, n-1\}$. The phase consists of q rounds numbered $0, 1, \ldots, q-1$. Then, in round i, we request repeatedly r^{h_i} until the state adjacent to J_i on the right, if $h_i = 0$, or on the left, if $h_i = 1$, acrues a cost of at least 1 in this round. Notice that the algorithm knows the entire round once its first request is given.

The analysis is completed as follows. Define

$$\bar{J}_i = \begin{cases} \{j : j \le \max s \in J_i\} & h_i = 0, \\ \{j : j \ge \min s \in J_i\} & h_i = 1. \end{cases}$$

When round i starts, with probability at least $\frac{1}{2}$, the algorithm occupies a state $s \notin \bar{J}_i$. In this case the algorithm pays at least 1 in this round. This is clear if it moves. Otherwise, every state not in $\bar{J}_i$ pays at least as much as the state in this set that is adjacent to J_i. Thus, the expected cost of the algorithm for the entire phase is at least $q/2 = \log(n-1)$. On the other hand, the adversary pays at most 1 to move to h. Notice that the state adjacent to J_i that determines the stopping condition pays less than 2 in round i (in fact it pays close to 1). Thus, all the states in J_i pay at most $\frac{2}{C}$. If the adversary stays at h for the duration of the phase, it pays a service cost of at most $\frac{2q}{C}$, which can be made arbitrarily close to 0 by choosing $C \gg q$. ◀

▶ **Lemma 11.** *Let $\mathcal{M}$ be an n-point uniform metric space. Then, $c^{\mathrm{det}}_{\mathcal{M},m} = \Omega(m \log(en/m))$.*

Proof Sketch. Without loss of generality, we may assume that m is an even number which is at most $n/2$, that $2n$ is divisible by m and that $2n/m$ is a power of 2. (Otherwise, eliminate some states by having their cost always ∞ and use fewer requests. A constant fraction of both states and requests remains.) So, partition the states into $m/2$ disjoint subsets of size $2n/m$ each. For each subset, define two new requests as per the proof of Lemma 10, and outside the subset have both requests cost 0. In each subset, use the adversary's strategy from the proof of Lemma 10. The requests according to this strategy will be used in the steps when the algorithm occupies a state in the subset. Clearly, while the algorithm hasn't paid at least $\Omega(\log(n/m))$ in all subsets, there's at least one state that hasn't paid 1. Thus, the adversary can choose the last such surviving state and cause the algorithm to pay $\Omega(m \log(n/m))$ before no state survives. When no state survives, a phase that cost the adversary less than 2 and the algorithm $\Omega(m \log(n/m))$ ends, and we can repeat the process in a new phase. ◀

Proof of Theorem 4. The theorem is a corollary of Lemmas 9, 10, and 11. ◀

Proof of Theorem 5. The upper bound on $c^{\text{rand}}_{\mathcal{M},2}$ follows from the general upper bound on trees of Bubeck et al. [13], and the lower bound follows from Theorem 4. The lower bound on $c^{\text{det}}_{\mathcal{M},2}$ goes as follows. Denote the pairs of leaves by i, i' for $i = 0, 1, \ldots, n/2 - 1$. Let $h_0, h_1, \ldots, h_{n/2-1}$ be the sequence defined by $h_i = (Cn)^{i-n}$. We use two requests: r is defined by $r_i = h_i$ and $r_{i'} = 0$, for all i, and r' is defined by $r'_i = 0$ and $r'_{i'} = h_{n-i-1}$, for all i. The request sequence is simple: if the algorithm occupies a state i, use r, otherwise the algorithm occupies a state i', so use r'.

For analyzing the competitive ratio, partition the request sequence into rounds. A round ends whenever the algorithm either moves from one pair to another pair or pays C while staying in one pair $\{i, i'\}$. Notice that either way, the algorithm's cost is at least C per round. Further partition the rounds into phases. Each phase contains exactly $n/2 - 1$ rounds. Thus, the cost of the algorithm per phase is at least $C(n/2 - 1)$. We show that the adversary can pay less than $C + n/2$ per round. Choosing $C = \Omega(n)$ completes the proof. To show that the adversary's cost per phase is less than $C + n/2$, notice that in a phase, the algorithm cannot visit all the pairs. Let $\{i, i'\}$ be a pair that is not visited by the algorithm during the phase. The adversary will "hide" in such a pair. Moving to this pair in the beginning of the round costs at most C. For each round in the phase, the adversary stays either at i or at i', so moving between i and i' costs at most $n/2 - 1$.

To complete the analysis, we have to explain how the adversary chooses between i and i', and we have to analyze the cost of staying at the chosen state. Notice that in a round, by definition the algorithm stays in one pair $\{j, j'\}$, where $i \neq j$. If $i < j$, the adversary stays at i, otherwise the adversary stays at i'. Consider the first case (the other case is analogous). By the definition of a round, j acrues a cost of at most C during the round, because it is hit by a positive cost only when the algorithm occupies j, and the algorithm pays at least this cost (it might decide to move to j' and pay 1). Therefore, i acrues a cost of at most $C \cdot (Cn)^{i-j} \leq \frac{1}{n}$. This is the service cost that the adversary pays during the round. Thus, since there are $n/2 - 1$ rounds in a phase, the total service cost of the adversary during a phase is less than 1. ◀

▶ **Lemma 12.** *Let $\mathcal{M}$ be an n-point uniform metric space. Then,* $\min\{m, n\} - 1 \leq \hat{c}^{\text{det}}_{\mathcal{M},m} \leq \min\{m, n\}$

Proof. The lower bound follows from restricting the number of states that can get a request with a 0 value to $k + 1 = \min\{m, n\}$ and then applying the deterministic k-server lower bound to this subspace. Notice that this lower bound uses $k + 1$ different requests.

The upper bound is achieved with a variant of the marking algorithm, as follows. Each request r can be associated with a subset $S_r = \{i \in X : \ r_i = 0\}$. If there exists a state i that is in the intersection of all m subsets S_r, the algorithm can pay at most 1 to move to this state, and then pay 0 for the rest of the sequence. Otherwise, the algorithm partitions the sequence into phases. In each phase, if the current location of the algorithm is hit by an ∞ value, it moves to a state in the intersection of all the sets seen so far in the phase. The phase ends, and a new phase begins, when this intersection is empty. Notice that to move, the algorithm needs to get a request not seen so far in the phase, so the algorithm moves at most m times per phase. At the end of the phase, all states were hit by an ∞ value at least once, so any algorithm would have to move at least once per phase. ◀

▶ **Lemma 13.** *Let $\mathcal{M}$ be an n-point uniform metric space. Then,* $\hat{c}^{\text{rand}}_{\mathcal{M},m} = O\left(\min\{m, \log n\}\right)$.

Proof. Theorem 3 implies in particular an upper bound of $O(\log n)$. Lemma 12 gives an upper bound of m (because the randomized competitive ratio is upper bounded by the deterministic competitive ratio). ◀

▶ **Lemma 14.** *Let $\mathcal{M}$ be an n-point uniform metric space. Then, $\hat{c}^{\mathrm{rand}}_{\mathcal{M},m} = \Omega\left(\min\{m, \log n\}\right)$.*

Proof. If $m \geq n$, then we get a lower bound of $\mathcal{H}_{n-1}$ from the paging problem [21], which requires n possible requests. Otherwise, let $k = \lfloor \min\{m/2, \log_2 n\} \rfloor$. We restrict our attention to a set of size 2^k states. (If there are more, all the others have cost ∞ in all the requests.) We label the states in this set with the nodes of the binary cube $\{0,1\}^k$. We use $2k$ possible requests that are constructed as follows. There are two requests $r_{i,0}$ and $r_{i,1}$ corresponding to each coordinate $i = 1, 2, \ldots, k$ of the binary cube. The request $r_{i,b}$ has a cost of ∞ at points with the i-th coordinate of their label being b, and 0 at states with the complement i-th coordinate. We use the minimax principle and construct a probabilistic request sequence that beats every deterministic algorithm. The request sequence consists of phases. In each phase the adversary draws independently a point $b_1 b_2 \cdots b_k$ in the binary cube uniformly at random, and then requests the sequence r_{i,b_i}, $i = 1, 2, \ldots, k$, in that order. Against any deterministic algorithm, the expected cost of each request in the phase is $\frac{1}{2}$, so the total expected cost of $\frac{k}{2}$. The adversary needs to move at most once per phase, to the point labeled with the bitwise complement of $b_1 b_2 \cdots b_k$. ◀

Proof of Theorem 6. The theorem is an immediate corollary of Lemmas 12, 13, and 14. ◀

▶ **Lemma 15.** *For every two-level HST $\mathcal{M}$, $\hat{c}^{\mathrm{det}}_{\mathcal{M},m} \leq m2^m$.*

Proof. Let $C > 1$ denote the aspect ratio of $\mathcal{M}$. Let T be any level-1 subtree of $\mathcal{M}$ (so in particular T indicates a uniform subspace of $\mathcal{M}$). We say that T is hit by a subset S of requests iff for every state $s \in T$ there is a request $r \in S$ such that $r_s = \infty$. The algorithm works in periods. In the beginning of a period all level-1 subtrees are unmarked, and a period ends when they are all marked. Each period is divided into epochs. In an epoch, the algorithm chooses an unmarked subtree T and stays in it for the entire epoch. While in a subtree T, the algorithm runs the uniform metric procedure from the proof of Lemma 12, ignoring the states outside T. Notice that this divides the epoch into phases, where during a phase the algorithm encounters one subset S of requests that hits T, for which the algorithm pays $|S| \leq m$ and any algorithm staying in T pays at least 1. The epoch ends as soon as the algorithm executes C phases in T. When the epoch ends, the algorithm marks T and any other subtree that was hit at least C times so far in the period. The cost of the algorithm in a period is upper bounded as follows. In each phase the algorithm encounters a hitting set S. If the same set is encountered C times, all the subtrees that it hits get marked. Therefore, since the empty set is not a hitting set, the total number of phases is at most $C(2^m - 1)$. The number of epochs in a period is therefore $2^m - 1$, and thus the total cost of transition between subtrees is $C(2^m - 2)$ (because no transition is required at the end of the last epoch in the period). Since the algorithm encounters each non-empty set S at most C times, the total "internal" cost of the phases is at most $C \cdot \sum_S |S| = Cm2^{m-1}$. Therefore, the total cost of the algorithm per period is $C(m2^{m-1} + 2^m - 2) < Cm2^m$, for all $m \in \mathbb{N}$. Any algorithm must pay at least C per period. If it stays in one subtree, then every subtree is hit at least C times. If it moves between subtrees, this incurs a cost of at least C. ◀

▶ **Lemma 16.** *For all $m \in \mathbb{N}$, $\hat{c}^{\mathrm{det}}_{2,m} \geq \binom{m}{\lfloor m/2 \rfloor}$.*

Proof. We construct the following two-level HST with aspect ratio C and request set $\mathcal{R}$. There are $\binom{m}{\lfloor m/2 \rfloor}$ subtrees, each labeled by a different subset of the requests of cardinality $\lfloor m/2 \rfloor$. Each subtree has $\lfloor m/2 \rfloor$ states labeled by the elements of the subtree's label. A request has cost ∞ at all the states that are labeled by it, and 0 otherwise. The adversary's strategy is simple: if the algorithm is currently at a subtree labeled S, apend to the sequence all the requests in S (the order doesn't matter), then repeat after updating the location that the algorithms reaches after the apended requests. In order to analyze the competitive ratio, partition the request sequence into periods, where in each period the algorithm leaves all the subtrees. Fix a period, and let k denote the number of times the algorithm changes a subtree in this period, including the departure that ends the period. Clearly, $k \geq \binom{m}{\lfloor m/2 \rfloor}$. The algorithm pays at least Ck in this period. The adversary can hide at the last tree that is visited. At each step, no matter which subtree the algorithm occupies, there is at least one state in the last tree that won't be hit by the apended requests. Each time the algorithm switches trees, the adversary must move to a new state in this last subtree. At the end of the period, the adversary may have to move to a new subtree. So, the total cost of the adversary per period is at most $k + C$. If we choose $C = \binom{m}{\lfloor m/2 \rfloor}$, this gives a lower bound of $\frac{1}{2} \cdot \binom{m}{\lfloor m/2 \rfloor}$. However, we can get a lower bound arbitrarily close to $\binom{m}{\lfloor m/2 \rfloor}$ by choosing a larger C. ◀

▶ **Lemma 17.** *For all $m \in \mathbb{N}$, $\hat{c}^{rand}_{2,m} \geq 2^{\lfloor m/2 \rfloor - 1}$.*

Proof. The argument is basically a "lifting" of the construction in the proof of Lemma 14. As in that proof, we associate the requests with the coordinates of the binary cube $\{0,1\}^{m/2}$ (we assume without loss of generality that m is even, otherwise, discard one request). Each state is labeled by a node of the cube, and request $r_{i,b}$ has cost ∞ for all states with a label that has value b in its i-th coordinate, and cost 0 otherwise. The requests are paired into pairs $r_{i,0}, r_{i,1}$. Notice that each request hits half of the possible state labels, and the sets of labels that two paired requests hit are complements. Now, consider a sequence of requests generated by choosing one request from each pair. We can label this sequence with a point b in the binary cube, where the sequence requests r_{i,b_i} for $i = 1, 2, \ldots, m/2$. Thus, there are $2^{m/2}$ possible such sequences. If such a sequence is requested, it hits all the labels, except for one – the bitwise complement of b. Thus, we can pair such sequences into pairs that have complement labels, and the two sequences in a pair miss complement labels of states. There are $2^{m/2-1}$ pairs of sequences. We can create a meta-sequence of requests by choosing one sequence in each pair and concatenating all the chosen sequences. This gives $2^{2^{m/2-1}}$ possible meta-sequences. This structure is used to generate the adversary's strategy.

But, before we state the adversary's strategy, we construct the metric space and set the labels of the states (which imply the structure of the individual requests). The two-level HST that we use has $2^{2^{m/2-1}}$ subtrees. A subtree is labeled by a subset of the binary cube $\{0,1\}^{m/2}$ of cardinality $2^{m/2-1}$ (half the cube) that contains exactly one node of each antipodal (i.e., complementary) pair of nodes. There are $2^{2^{m/2-1}}$ possible choices, hence the number of subtrees. Such a subtree has $2^{m/2-1}$ states, each labeled with a distinct node in the label of the subtree. Aside from the aspect ratio C, to be defined later, this specifies completely $\mathcal{M}$ and $\mathcal{R}$.

We are now ready to define the adversary's strategy. As usual, we rely on the minimax principle and define a randomized strategy that beats any deterministic algorithm. The adversary repeats the following process. Choose one meta-sequence uniformly at random, then loop through its list of sequences and repeat each sequence C times. Notice that regardless of the state of the algorithm at the beginning of any sequence of the meta-sequence, it pays at least C for this sequence with probability at least $\frac{1}{2}$. This is because each sequence

hits half the subtrees, and the two possible choices for the sequence hit complement sets of subtrees: a sequence labeled b hits all subtrees that contain b, a sequence labeled $\bar{b}$ hits all subtrees that contain $\bar{b}$, and every subtree contains either b or $\bar{b}$, regardless of b. If the sequence hits the subtree where the algorithm is located, it either stays there and pays at least 1 for each time the sequence is requested (because it hits all the states in that subtree), or it moves to a different tree and pays C for the transition. On the other hand, after requesting the entire meta-sequence, there exists one subtree that was not hit even once. This is the subtree that is labeled with the set of complements of all the labels of the sequences in the meta-sequence. The adversary can hide in that subtree. It pays C to move there at the start of the current meta-sequence, and at most 1 to move to a safe state at the start of each sequence in the meta-sequence. So, the expected cost of the algorithm per meta-sequence is $C2^{m/2-1}$, and the total cost of the adversary per meta-sequence is $C + 2^{m/2-1}$. As C grows, the ratio approaches $2^{m/2-1}$. ◀

Proof of Theorem 7. The theorem is a corollary of Lemmas 15 and 17. ◀

4 Discussion

4.1 Open problems

Our work initiates the study of metrical task systems (MTS) and metrical service systems (MSS) parametrized by the number of distinct requests. Roughly speaking (and somewhat surprisingly) we find that this restriction has little effect in general for MTS, in the sense that beyond uniform metric spaces, the achievable competitive ratio with $m = 2$ is already asymptotically as bad as for $m = \infty$ (Theorem 5). In fact, as far as randomized algorithms are concerned this is already true for uniform metric spaces (Theorem 4). On the other hand, the situation for MSS is very different. A number of questions remain open regarding MSS. A particularly interesting qualitative open problem would be to characterize the class of infinite size metric spaces for which there exist online MSS algorithms with finite competitiveness for fixed m, say even for $m = 3$. For example, is it possible to obtain a finite competitive ratio for chasing 3 arbitrary sets on the real line? Another intriguing quantitative open problem is to determine if the deterministic competitive ratio on a weighted star metric is linear or exponential in m. (A simple argument bounds the deterministic competitive ratio by $O(2^m)$ and the randomized competitive ratio by $O(m)$. A lower bound of $\Omega(m)$ in both cases clearly follows from Theorem 6.)

4.2 Deeper trees

Consider MSS on an HST. Label each leaf by a binary vector of dimension m indicating which requests hit it. Clearly, if there are two identically labeled leaves that share a parent, we can eliminate one from consideration. Similarly, label each internal node by the set of labels of its children. If there are two identically labeled internal nodes with the same label and the same parent, we can eliminate one. Thus, effectively, the maximum number T_L of distinct L-level HSTs satisfies: $T_0 < 2^m$ and $T_L < 2^{T_{L-1}}$ (the reason for the strict inequality is that we can eliminate the empty tree and also every leaf labeled by the all-ones vector). The maximum number of leaves N_L of an L-level HST therefore satisfies $N_L \le \prod_{i=0}^{L-1} T_i$ (in fact, this estimate is far from tight for large L). Notice that $\log_2 N_L = \sum_{i=0}^{L-1} \log_2 T_i = O(\log_2 T_{L-1})$. Theorem 7 can be generalized to any number of levels $L \ge 2$. Let $c_1 = \lfloor m/2 \rfloor$ and for $L \ge 2$ let $c_L = 2^{c_{L-1}-1}$. Then, for $L \ge 1$,

$$\Omega(c_L) \le \hat{c}^{\mathrm{rand}}_{L,m} \le O(\log N_L).$$

The upper bound follows from the above observations and previous work (Theorem 3). As for the lower bound, in the proof of Lemma 17 we constructed $2^{2^{m/2-1}}$ one-level subtrees, each labeled with a subset of the binary cube $\{0,1\}^{m/2}$ of cardinality $2^{m/2-1}$, and also $2^{2^{m/2-1}}$ meta-sequences. Each meta-sequence misses one subtree. The subtrees can be paired into complement halves of the binary cube, and the meta-sequences can be paired into complement choices of the sequences that compose them. Complement meta-sequences miss complement subtrees. Thus, we can "lift" this construction just as we "lifted" the uniform metric construction to get a $2^{2^{m/2-1}-1}$ randomized lower bound for three-level HSTs, and this "lifting" can be iterated ad infinitum.

For three or more levels, we do not know reasonably tight upper and lower bounds on the deterministic competitive ratio. We leave this as an open problem. For two levels, the upper and lower bounds of Theorem 7 are similar but not tight asymptotically in m. Thus, deriving asymptotically tight bounds, and moreover determining if the deterministic and randomized competitive ratios are asymptotically (in m) equal (as is the case for uniform metric spaces), are interesting open problems.

4.3 Leaky randomization

Generally our results show that the characteristics of the metric spaces have a strong influence on the type of guarantees one can hope for, in stark contrast with the well-known results and conjectures for non-parametrized sequences (e.g., randomized MTS, or the k-server problem). Moreover our lower bounds constructions also shed a new light on randomization, in the following sense. In online computing randomness may help because it hides the state of the algorithm. An adversary generating the worst-case sequence for a given algorithm knows the probability distribution but not the outcome of the algorithm's coin flips. A natural question in this context is to quantify this phenomenon. In particular, consider an adversary that is given at each step t a signal σ_t indicating some information on the algorithm's state s_t (which is a random variable). Suppose that there exists b such that for all t, the mutual information is $I(\sigma_t; s_t) = b$. If the adversary is allowed to choose the signals subject to this constraint, what can we say about the competitive ratio? Before we proceed, two comments are in place. Firstly, by "algorithm's state" we could mean simply the position reached by the algorithm in the state space, or we could mean more broadly also the internal state of the algorithm. The distinction is immaterial for our results. Secondly, notice that such an adversary is restricted even from having perfect recall, because past requests may reveal more than b bits of information regarding the algorithm's current state. So, we denote the best competitive ratio against the above adversary by $c^b_{\mathcal{M}}$. In every metric space $\mathcal{M}$, the deterministic lower bound in Theorem 1 implies that $c^{\lceil \log n \rceil}_{\mathcal{M}} = 2n - 1$. Notice that $c^{\det}_{\mathcal{M},2^b}$ reveals something about $c^b_{\mathcal{M}}$. In particular, in paired-uniform metric spaces, revealing a single bit is sufficient to force the algorithm to pay asymptotically as much as a deterministic algorithm.

▶ **Corollary 18.** *If $\mathcal{M}$ is a paired-uniform metric space, then*

$$c^1_{\mathcal{M}} = \Omega(n).$$

Proof of Corollary 18. If $c^{\det}_{\mathcal{M},m} = \Omega(n)$, then in order to force an algorithm to a competitive ratio of $\Omega(n)$, all that the adversary needs to know is which of the m requests to use at each step. For this $\log_2 m$ bits of information are sufficient, and this is clearly an upper bound on the mutual information. Thus, the corollary follows from Theorem 5. ◀

References

1 J. Abernethy, P. L. Bartlett, N. Buchbinder, and I. Stanton. A regularization approach to metrical task systems. In *Proc. of the 21st Int'l Conf. on Algorithmic Learning Theory*, pages 270–284, 2010.

2 J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. *SIAM J. Comput.*, 37(5):1499–1516, 2008.

3 R. A. Baeza-Yates, J. C. Culberson, and G. J. E. Rawlins. Searching in the plane. *Inf. Comput.*, 106(2):234–252, 1993.

4 N. Bansal, N. Buchbinder, and J. Naor. Metrical task systems and the k-server problem on HSTs. In *Proc. of the 37th Int'l Colloq. on Automata, Languages and Programming*, pages 287–298, 2010.

5 Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In *Proc. of the 37th Ann. IEEE Symp. on Foundations of Computer Science*, pages 184–193, 1996.

6 Y. Bartal, A. Blum, C. Burch, and A. Tomkins. Polylog(n)-competitive algorithm for metrical task systems. In *Proc. of the 29th Ann. ACM Symp. on Theory of Computing*, pages 711–719, 1997.

7 Y. Bartal, B. Bollobas, and M. Mendel. A Ramsey-type theorem for metric spaces and its applications for metrical task systems and related problems. In *Proc. of the 42nd Ann. IEEE Symp. on Foundations of Computer Science*, pages 396–405, 2001.

8 Y. Bartal, M. Charikar, and P. Indyk. On page migration and other relaxed task systems. In *Proc. of the 8th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 43–52, 1997.

9 Y. Bartal, N. Linial, M. Mendel, and A. Naor. On metric Ramsey-type phenomena. In *Proc. of the 35th Ann. ACM Symp. on Theory of computing*, pages 463–472, June 2003.

10 A. Blum and C. Burch. On-line learning and the metrical task system problem. In *Proc. of the 10th Ann. Conf. on Computational Learning Theory*, pages 45–53, 1997.

11 A. Blum, H. J. Karloff, Y. Rabani, and M. E. Saks. A decomposition theorem for task systems and bounds for randomized server problems. *SIAM J. Comput.*, 30(5):1624–1661, 2000.

12 A. Borodin, N. Linial, and M. E. Saks. An optimal on-line algorithm for metrical task system. *J. ACM*, 39(4):745–763, 1992.

13 S. Bubeck, M. B. Cohen, J. R. Lee, and Y.-T. Lee. Metrical task systems on trees via mirror descent and unfair gluing. In *Proc. of the 30th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 89–97, 2019.

14 W. R. Burley. Traversing layered graphs using the work function algorithm. *J. Algorithms*, 20(3):479–511, 1996.

15 W. R. Burley and S. Irani. On algorithm design for metrical task systems. *Algorithmica*, 18(4):461–485, 1997.

16 A. Chiplunkar and S. Vishwanathan. Metrical service systems with multiple servers. *Algorithmica*, 71(1):219–231, 2015.

17 M. Chrobak and L. Larmore. Metrical service systems: deterministic strategies. Technical Report UCR-CS-93-1, UC Riverside, 1993.

18 J. Fakcharoenphol, S. Rao, and K. Talwar. A tight bound on approximating arbitrary metrics by tree metrics. *J. Comput. Syst. Sci.*, 69(3):485–497, 2004.

19 E. Feuerstein. Uniform service systems with k servers. In Cláudio L. Lucchesi and Arnaldo V. Moura, editors, *Proc. of LATIN'98: Theoretical Informatics*, pages 23–32, 1998.

20 A. Fiat, D. P. Foster, H. Karloff, Y. Rabani, Y. Ravid, and S. Vishwanathan. Competitive algorithms for layered graph traversal. In *Proc. of the 32nd Ann. IEEE Symp. of Foundations of Computer Science*, pages 288–297, 1991.

21 A. Fiat, R. M. Karp, M. Luby, L. A. McGeoch, D. D. Sleator, and N. E. Young. Competitive paging algorithms. *J. Alg.*, 12(4):685–699, 1991.

22 A. Fiat and M. Mendel. Better algorithms for unfair metrical task systems and applications. In *Proc. of the 32nd Ann. ACM Symp. on Theory of computing*, pages 725–734, May 2000.

23 J. Friedman and N. Linial. On convex body chasing. *Discrete & Computational Geometry*, 9(3):293–321, 1993.

24 S. Irani and S. Seiden. Randomized algorithms for metrical task systems. *Theoretical Computer Science*, 194(1):163–182, 1998.

25 S. Irani, S. Shukla, and R. Gupta. Online strategies for dynamic power management in systems with multiple power-saving states. *ACM Trans. Embed. Comput. Syst.*, 2(3):325–346, 2003.

26 H. J. Karloff, Y. Rabani, and Y. Ravid. Lower bounds for randomized k-server and motion-planning algorithms. In *Proc. of the 23rd Ann. ACM Symp. on Theory of Computing*, pages 279–288, May 1991.

27 M. S. Manasse, L. A. McGeoch, and D. D. Sleator. Competitive algorithms for server problems. *J. Algorithms*, 11(2):208–230, 1990.

28 C. H. Papadimitriou and M. Yannakakis. Shortest paths without a map. In *Proc. of the 16th Int'l Colloq. on Automata, Languages and Programming*, pages 610–620, 1989.

29 H. Ramesh. On traversing layered graphs on-line. In *Proc. of the 4th Ann. ACM-SIAM Symp. on Discrete Algorithms*, pages 412–421, 1993.

30 S. Seiden. Unfair problems and randomized algorithms for metrical task systems. *Inf. Comput.*, 148(2):219–240, 1999.

31 D. D. Sleator and R. E. Tarjan. Amortized efficiency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

A Constant Factor Approximation for Capacitated Min-Max Tree Cover

Syamantak Das
IIIT Delhi, India
http://faculty.iiitd.ac.in/~syamantak/
syamantak@iiitd.ac.in

Lavina Jain
IIIT Delhi, India
Google scholar page
lavina16052@iiitd.ac.in

Nikhil Kumar
IIT Delhi, India
Google scholar page
nikhil@cse.iitd.ac.in

Abstract

Given a graph $G = (V, E)$ with non-negative real edge lengths and an integer parameter k, the (uncapacitated) Min-Max Tree Cover problem seeks to find a set of at most k trees which together span V and each tree is a subgraph of G. The objective is to minimize the maximum length among all the trees. In this paper, we consider a capacitated generalization of the above and give the first constant factor approximation algorithm. In the capacitated version, there is a hard uniform capacity (λ) on the number of vertices a tree can cover. Our result extends to the rooted version of the problem, where we are given a set of k root vertices, R and each of the covering trees is required to include a distinct vertex in R as the root. Prior to our work, the only result known was a $(2k-1)$-approximation algorithm for the special case when the total number of vertices in the graph is $k\lambda$ [Guttmann-Beck and Hassin, *J. of Algorithms, 1997*]. Our technique circumvents the difficulty of using the minimum spanning tree of the graph as a lower bound, which is standard for the uncapacitated version of the problem [Even et al.,*OR Letters 2004*] [Khani et al.,*Algorithmica 2010*]. Instead, we use *Steiner trees* that cover λ vertices along with an iterative refinement procedure that ensures that the output trees have low cost and the vertices are well distributed among the trees.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Approximation Algorithms, Graph Algorithms, Min-Max Tree Cover, Vehicle Routing, Steiner Tree

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.55

Category APPROX

1 Introduction

Covering vertices of a given graph using simpler structures, for example, paths, trees, stars and so on, have long attracted the attention of the Computer Science and Operations Research communities. This can be attributed to a variety of applications in vehicle routing, network design and related problems. One classical example is the so-called "Nurse Location Problem" [10]. The goal is to place a group of nurses at different locations and finding a tour for each of them so that every patient is visited by a nurse. A similar setting arises in vehicle routing. Suppose we are given a set of vehicles, initially located at a given set of depots. The goal is to find a tour for each of these vehicles, each starting and ending at the respective depots so as to cover client demands at various locations. One of the most popular objectives is to minimize the maximum distance travelled by any vehicle, also known as the


© Syamantak Das, Lavina Jain, and Nikhil Kumar;
licensed under Creative Commons License CC-BY



makespan of the solution. A standard reduction shows that this problem is equivalent, within an approximation factor of 2, to finding trees in a graph such that all vertices in the graph are covered by the union of these trees. In this paper, we consider a natural generalization of the above setting. As before, we are given a set of vehicles, initially located at a given set of depots and a set of clients. Additionally one package is to be delivered to each of the clients and each vehicle can carry at most a fixed number of packages (all packages are identical). We have to find a set of tour such that each client receives a package and the objective, as before is to minimize the maximum distance travelled by any vehicle. This can be seen as a capacitated version of the Nurse Location Problem, where each nurse can visit at most a given number of patients. Within a factor of two, this problem is equivalent to the following generalization of the TSP problem : given a metric and two constants k, λ, find k tours, each tour on no more than λ vertices, such that the maximum cost over all the tours is minimized. We formally define the problem now, which will be useful in further discussions.

1.1 Notations and Preliminaries

We set up some preliminaries, notations and definitions from literature that will be useful in the exposition of our contributions. The set of positive integers $\{1, 2, , \ldots, n\}$ is denoted by $[n]$. Given a graph $G = (V, E)$, $H = (V_H, E_H)$ is a subgraph of G if $V_H \subseteq V$ and $E_H \subseteq E$. We use $\ell(H)$ to denote the total length of edges in the subgraph H. For any two vertices u, v, $d(u, v)$ denotes the shortest path distance between u, v. We extend the definition to subsets of vertices U, V - define $d(U, V)$ to be $\min_{u \in U, v \in V} d(u, v)$. The set of vertices in any graph H is denoted by $V(H)$.

▶ **Definition 1** (Min-Max Tree Cover Problem (MMTC)). *Given a graph $G(V, E)$, edge length $\ell : E \to \mathbb{R}_{\geq 0}$ and a parameter k, one is required to output a set of k trees T_i for $i \in [k]$, such that each T_i is a subgraph of G and $\cup_{i=1}^{k} V(T_i) = V$. The objective is to minimize $max_{i=1}^{k} \ell(T_i)$.*

Note that two trees in a feasible solution can share vertices as well as edges. In the capacitated version of the problem, we are also given an additional parameter λ. A feasible solution to the Capacitated Min-Max k-Tree Cover consists of a set of trees (not necessarily disjoint) $\mathcal{T} = \{T_1, T_2, \ldots, T_k\}$ along with an assignment of each vertex $v \in V$ to one of the trees containing it, such that no more than λ vertices are assigned to any $T \in \mathcal{T}$. The goal, as in the case of uncapacitated case, is to minimize the maximum length of any tree in $\mathcal{T}$. Let $\text{cov}(T)$ denote the set of vertices assigned to T. We think of the vertices in $\text{cov}(T)$ as being covered by T. Hence, in any feasible assignment $\cup_{T \in \mathcal{T}} \text{cov}(T) = V$ and $|\text{cov}(T)| \leq \lambda$ for all $T \in \mathcal{T}$. Note that a vertex may be a part of multiple trees, but it is covered by exactly one of them. Whenever the notation $\text{cov}(T)$ is used, there is an (implicit) underlying assignment of vertices to the trees. We do not explicitly mention the assignment if it is clear from the context.

▶ **Definition 2** (Capacitated Min-Max Tree Cover (CapMMTC)). *Given a graph $G(V, E)$, edge length $\ell : E \to \mathbb{R}_{\geq 0}$, and two integer parameters k, λ, one is required to output a set of k trees T_i for $i \in [k]$ along with an assignment of every vertex in G to a tree containing it, such that each T_i is a subgraph of G, $\cup_{i=1}^{k} \text{cov}(T_i) = V$ and $|\text{cov}(T_i)| \leq \lambda$ for $i \in [k]$. The objective is to minimize $max_{i=1}^{k} \ell(T_i)$.*

In rooted version of MMTC and CapMMTC, we are given a set of k roots as well. The only additional constraint being that each tree in the output should contain a distinct root. We will refer to these problems as Rooted Min-Max Tree Cover Problem(RMMTC) and Capacitated Rooted Min-Max Tree Cover (CapRMMTC).

1.2 Our Contribution

We give the first polynomial time constant factor approximation algorithm for CapMMTC and CapRMMTC.

▶ **Theorem 3.** *There is a polynomial time $\mathcal{O}(1)$-approximation algorithm for* CapMMTC.

▶ **Theorem 4.** *There is a polynomial time $\mathcal{O}(1)$-approximation algorithm for* CapRMMTC.

Our algorithms are much more intricate and involved than those which give an $\mathcal{O}(1)$ approximation for the uncapacitated case, as is generally the case with capaciated versions of many problems in combinatorial optimization, for instance k-median and facility location problems. These are the first approximation algorithms for both the problems, to the best of our knowledge. The only known result is a $(2k-1)$-approximation for the special case when the total number of vertices in the graph is $k\lambda$ and hence every tree must contain exactly λ vertices [12]. All our algorithms are combinatorial. We first show the result for the unrooted case, ie. CapMMTC and then extend the ideas to prove the result for the rooted problem CapRMMTC. We show that value of the constant in Theorem 3 and 4 is ≤ 300. For simplicity of exposition, we have not tried to optimize the constant. We believe that it should be possible to do so with some more effort and leave it as an open problem. We prove Theorem 3 in Section 2 and defer the proof of Theorem 4 to the Appendix, Section 3.

1.3 Related Work

Even et al. [10] and Arkin et al. [3] gave 4-approximation algorithms for both rooted and unrooted (uncapacitated) MMTC. Khani et al. improved the unrooted version to a 3-approximation [16], whereas Nagamochi and Okada [17] gave a $3-2/(k+1)$-approximation for the special case of the rooted version where all roots are co-located. On the other hand, the MMTC problem has been shown to be hard to approximate to a factor better than $3/2$, assuming $P \neq NP$ [20]. The problem has been considered when the underlying graph has special structure. The rooted version of the MMTC problem on a tree admits a PTAS, as shown recently by Becker and Paul [5], while the unrooted version has a $(2+\epsilon)$-approximation, given by Nagamochi and Okada [17]. Further, Chen and Marx gave fixed parameter tractable algorithms for the problem on a tree [8]. MMTC on a star is equivalent to the classical makespan minimization problem on identical machine, for which elegant EPTAS-es are well known [1]. In stark contrast, no approximation algorithms have been reported so far for the capacitated versions of these problems, to the best of our knowledge. Guttmann-Beck and Hassin gave an $(2k-1)$-approximation for the special case where the number of vertices in each the tree is exactly λ [12]. On the other special case of a star metric, an EPTAS result follows from the work on identical machine scheduling with capacity constraints [7]. Interestingly, the problem becomes inapproximable if one disallows sharing of vertices and edges between the trees in the solution [12], even when $k = 2$. A related but quite different setting is that of bounded capacity vehicle routing with the objective of minimizing the total length of all the tours. Here, a vehicle is allowed to make multiple tours to cover all the points, however, each tour can serve at the most λ clients. This problem has been well studied, a $2.5 - \lambda^{-1}$-approximation is known for general graphs [13], while Becker et al. have reported a PTAS on planar graphs recently [4]. Capacitated versions of other classical combinatorial optimization problems are very well studied and we give a highly non-exhaustive list here. Capacitated unweighted vertex cover admits a 2-approximation [14, 19]. Capacitated versions of clustering problems like k-center [2], k-median [6] and scheduling problems [18] have also been widely popular.

1.4 High Level Ideas and Techniques

Most algorithms for either MMTC or rooted MMTC build upon the following idea. Assume that we know the optimal tree cost - say $T^\star$. Further, for ease of exposition, let us assume that the graph is connected and does not contain any edge of length more than $T^\star$. Now consider the k trees in the optimal solution, each of total length at most $T^\star$. Adding at most $k-1$ edges to the union of these trees forms a spanning tree of the entire graph. Hence, the minimum spanning tree (MST) cost is upper bounded by $(2k-1)T^\star$. This leads to the natural idea of starting with the MST of the graph. At a high-level, we can root the tree at an arbitrary vertex, traverse it bottom-up and chop it off as soon as the total cost of the traversed sub-tree is between $[2T^\star, 4T^\star]$. This needs to be done carefully and we would refer the reader to [10] for details. Now, this can create at most k partitions of the MST, each of cost at most $4T^\star$. The rooted version uses a similar idea, but requires more care. Let us try to apply this idea to the capacitated case. A potential problem is that, we have no control over the structure of the MST. In particular, some part of the MST might be dense - it might contain a connected subtree that has small length but covers a large number of vertices, while some other parts might be sparse. Hence, cutting off the MST on the basis of length as above might end up producing infeasible trees, although the cost might be bounded. One possible idea to fix this could be to further cut off the dense subtrees and try and re-combine them with the sparse subtrees. However, it is not clear how to avoid either combining more than a constant number of subtrees or subtrees that are more than $\mathcal{O}(T^\star)$ distance away from each other and hence cannot lead to a constant factor approximation.

Our Approach: Using λ-Steiner Trees. In order to develop the intuition for our core ideas, we focus on the special case where each tree in the optimal solution *covers* exactly λ vertices (note that each tree may span more than λ vertices). We take a different approach to the problem by utilizing the concept of λ-Steiner Trees. Given a graph G and a subset of vertices R called *terminals*, a λ-Steiner tree on R is a minimum length subtree of G that contains exactly λ vertices from R. We begin with the observation that each tree in the optimal solution covers λ vertices and hence OPT can be thought of as an union of k λ-Steiner trees in G, although not necessarily of the minimum possible cost. A natural algorithm is to pick an arbitrary root vertex and construct a λ-Steiner tree. Computing the λ-MST and hence λ-Steiner trees is NP-Hard and hence we resort to the 4-approximation algorithm that essentially follows from Garg's algorithm [11]. If we are lucky, we might end up capturing an optimal tree and continue. However, in the unlucky case, the λ-Steiner tree might cover vertices from several of the optimal trees. Now if we disregard these vertices in further iterations and try to build another λ-Steiner Tree on the uncovered vertices, we might be stuck since such vertices in the union of the optimal trees might be far away from each other. Hence we cannot guarantee that a low cost λ-Steiner tree exists on these vertices.

We fix this problem by being less aggressive in the first step. We try to build as many $\lceil \lambda/2 \rceil$-Steiner trees as possible that have cost at the most $\mathcal{O}(T^\star)$ using Garg's algorithm - call such trees *good*. At some point, we might be left with vertices such that there does not exist any $\lceil \lambda/2 \rceil$-Steiner trees of low cost that can cover them - let us call them *bad vertices*. In order to cover the bad vertices, we deploy an iterative clustering procedure. We begin by applying the algorithm of Khani and Salavatipour [16] for MMTC, henceforth termed as the KS-algorithm, on the bad vertices. Note that this will return at most k trees each of cost $\mathcal{O}(T^\star)$, although we still cannot prove any lower bound on the number of vertices covered by each tree. Next we set up a bipartite matching instance with these newly formed trees on the left hand side and the good trees on the right hand side. We introduce an edge

between two trees if and only if they are separated by a distance of at the most $T^\star$. The crucial claim now is - if there exists a Hall Set in this matching instance, say S, then the number of trees in S is strictly greater than the number of trees that the optimal solution forms with the vertices covered by S. Hence, applying the KS-algorithm on the vertices in S reduces the number of trees in S without increasing the cost of each tree. This idea forms the heart of our algorithm. We apply the KS-algorithm iteratively until there is no Hall Set, at which point we can compute a perfect matching of the bad trees. Each bad tree can now be combined with a good tree to produce a tree that has cost $\mathcal{O}(T^\star)$ and contains at least $\lambda/2$ vertices each. In a nutshell, the above procedure circumvents the problem of creating too many sparse trees. It ensures that every tree is sufficiently dense - covers at least $\lceil \lambda/2 \rceil$ vertices and are of low cost. Consider a modified graph $\tilde{G}$ created by contracting the edges of the dense trees. Since the original graph is connected and each edge has length at the most $T^\star$, $\tilde{G}$ is also connected and any edge in $\tilde{G}$ has length at the most $T^\star$. We utilize these properties to distribute the vertices and ensure that the final set of trees have exactly λ vertices each and cost at the most $\mathcal{O}(T^\star)$. We note that under the assumption that every optimal solution tree has exactly λ vertices as well, the above algorithm will produce exactly k trees.

Our algorithm for the capacitated problem builds upon the above ideas. However, one major bottleneck is that we can no longer assume that each tree in the optimal solution contains exactly λ vertices. In fact, there could be trees which have a small number of vertices, say less than $\lambda/2$ - call them *light* trees and the rest *heavy*. Handling this situation requires more subtle ideas. We again start by creating $\lceil \lambda/4 \rceil$-Steiner trees of low cost as long as possible which we call good trees and as before, we shall be left with some bad trees that contain less than $\lambda/4$ vertices each, but have bounded cost. Unfortunately, the iterative refinement procedure is no longer guaranteed to produce a perfect matching, as before. However, we are able to show the following. Existence of a Hall Set even after applying the said refinement is a certificate of the fact that optimal solution contains a significant number of light trees - suppose this number is k_ℓ. Then, we add $k_\ell \cdot \lceil \lambda/2 \rceil$ dummy vertices co-located with suitably chosen bad vertices. The upshot is that, this addition ensures that each bad tree now becomes a good tree, together with the dummy vertices. Further, the total number of vertices including the dummy vertices is still bounded above by $k\lambda$. Together, this gives us that, creating trees of size exactly λ cannot result in more than k trees in the solution.

2 Capacitated Min-Max Tree Cover

In this section, we shall describe our algorithm for the CapMMTC problem and prove Theorem 3. The first step in our algorithm is to guess the value of the optimal solution - call it $T^\star$. We remove all edges from G that are of length bigger than $T^\star$ since the optimal solution can never use any such edge. The resulting graph $\hat{G}$ has, say, p connected components - call them $G_1, G_2, \cdots G_p, p \leq k$. Suppose, in the optimal solution, there are k_i trees that cover all vertices in $G_i, i = 1, 2 \cdots p$. Then, $\Sigma_{i=1}^{p} k_i = k$. For each connected component i, we run our algorithm to get at most k_i trees with cost $\mathcal{O}(T^*)$. Due to the above argument, in subsequent exposition, we shall assume that we have a connected graph $G(V, E)$ with edge lengths $\ell(e) \leq T^\star, \forall e \in E$ and there are k trees in the optimal solution that cover V such that every tree has cost at most $T^\star$ and covers at most λ vertices. There could be multiple optimal solutions. For the purpose of our discussion we pick one arbitrarily and whenever we refer to an optimal solution, we mean this particular solution. We shall divide the trees in the optimal solution into two classes for the purpose of analysis. Define a tree T in the optimal solution to be *light* if $|\text{cov}(T)| \leq \lfloor \lambda/2 \rfloor$ and *heavy* otherwise. Define k_{heavy} and k_{light} to be the number of heavy and light trees respectively. We shall prove the following theorem.

▶ **Theorem 5.** *Given a connected graph $G(V, E)$ with the edge lengths $\ell : E \to \mathbb{R}_{\geq 0}$, $\ell(e) \leq T^\star, \forall e \in E$ and non-negative integers k, λ, suppose there exists k trees $T_1, T_2, \ldots, T_k$ along with an assignment of vertices to the trees, such that T_i is a subgraph of G, $|\text{cov}(T_i)| \leq \lambda$ and $\ell(T_i) \leq T^\star$ for $i \in [k]$. Then there exists a polynomial time algorithm that finds a set of trees $T'_1, T'_2, \cdots T'_{k'}$ along with an assignment of vertices to the trees such that $k' \leq k$, for each T'_j, $\ell(T'_j) = \mathcal{O}(T^\star)$ and $|\text{cov}(T'_j)| \leq \lambda$.*

Before proceeding to the proof of the Theorem above, we show how it implies a proof for Theorem 3.

Proof of Theorem 3. We can use Theorem 5 to carry out a binary search for the correct value of $T^\star$ - the optimal solution - in the range $[0, \Sigma_{e \in E} \ell(e)]$. For a particular choice of $T^\star$, we remove all edges that are of length more than $T^\star$ and apply Theorem 5 to each of the connected components. If the total number of trees formed by our algorithm over all components is at most k, then we iterate with a guess $T^\star/2$, otherwise with $2T^\star$. The correctness follows from Theorem 5, since our algorithm will create at most k' trees for connected component V', provided optimal solution makes k' trees as well and the guessed value $T^\star$ is correct. ◀

▶ **Definition 6.** *Given an integer $\lambda > 0$, a tree T is called λ-good if $|\text{cov}(T)| \geq \lambda/4$.*

▶ **Definition 7** (λ-Steiner Trees)**.** *Given a graph $G(V, E)$ and a subset of vertices R called terminals, λ-Steiner tree is a tree which is a subgraph of G that spans exactly λ terminals. The special case of $R = V$ is called λ-MST.*

We shall be using the 2-approximation algorithm for minimum cost λ-MST from [11] as a black-box, which implies a 4-approximation for the minimum cost λ-Steiner tree problem. Our algorithm has three phases $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$, each making progress towards the proof of Theorem 5.

2.1 Algorithm $\mathcal{A}$: Constructing λ-good trees

Recall that the input is a graph $G = (V, E)$, with edge lengths $l_e \leq T^\star, e \in E$. Further, we assume that there exists a partition of V into k trees such that the cost of each tree is at most $T^\star$ and each tree covers at most λ vertices (with respect to some assignment). We maintain a set of covered vertices V_c throughout our algorithm. Initially $V_c = \phi$. The following is done iteratively. Pick an arbitrary vertex $v \in V \setminus V_c$ which has not been considered in any previous iteration. Construct a 4-approximate minimum cost $\lceil \lambda/4 \rceil$-Steiner Tree in G, rooted at v with the terminal set being $V \setminus V_c$. If the cost of this tree is at most $4T^\star$, then add this tree to the set λ_{good}, add all terminals spanned by this tree to V_c. At the termination of this loop, set $V_{bad} = V \setminus V_c$.

▶ **Lemma 8.** *At termination of Algorithm $\mathcal{A}$,*
1. *$V = \text{cov}(\lambda_{good}) \cup V_{bad}$, where $\text{cov}(\lambda_{good}) = \bigcup_{T \in \lambda_{good}} \text{cov}(T)$*
2. *For any tree $T \in \lambda_{good}$, $|\text{cov}(T)| = \lceil \lambda/4 \rceil$ and $\ell(T) \leq 4T^\star$*

▶ **Lemma 9.** *For any heavy tree in* OPT*, say T, at least half of the vertices covered by T in* OPT*, ie.* $\text{cov}(T)$ *are covered by the union of trees in λ_{good}*

Proof. Assume otherwise. This implies that at least $\lambda/4$ vertices covered by T are added to V_{bad} - let us call this set V'. Now, $\ell(T) \leq T^\star$ and there must exist a connected sub-tree of T that covers exactly $\lceil \lambda/4 \rceil$ vertices from V'. Hence, some iteration of Algorithm $\mathcal{A}$ must have returned a tree of cost at most $4T^\star$ covering $\lceil \lambda/4 \rceil$ vertices from V' and this tree was added to λ_{good}. Consequently, V' cannot be a subset of V_{bad} leading to a contradiction. ◀

Algorithm 1 Algorithm $\mathcal{A}$.

Input: Graph $G = (V, E), \lambda$
Output: Set of trees λ_{good} and set of vertices V_{bad}
$V_c \leftarrow \phi$, $\lambda_{good} \leftarrow \phi$, $V_{temp} \leftarrow V$
While $V_{temp} \neq \phi$
Pick $v \in V_{temp}$ arbitrarily
$T \leftarrow \lceil * \rceil \lambda/4$-Steiner tree rooted at v with terminal set defined as $V \setminus V_c$
$\text{cov}(T) \leftarrow$ terminals in $V \setminus V_c$ covered by T
If $\ell(T) \leq 4T^\star$ **Then**
Add T to λ_{good}
Update $V_{temp} \leftarrow V_{temp} \setminus \text{cov}(T), V_c \leftarrow V_c \cup \text{cov}(T)$
Else
Update $V_{temp} \leftarrow V_{temp} \setminus \{v\}$
$V_{bad} \leftarrow V \setminus V_c$
Return λ_{good} and V_{bad}

2.2 Algorithm $\mathcal{B}$: Covering V-bad vertices with λ-good trees

Recall that in the previous section, we gave algorithm to construct a set of trees λ_{good} such that for any tree $T \in \lambda_{good}$, $\ell(T) \leq 4T^\star$ and $|\text{cov}(T)| = \lceil \lambda/4 \rceil$. Further, we extracted a subset of vertices V_{bad} such that no $\lceil \lambda/4 \rceil$-Steiner tree of cost at the most $4T^\star$ exists covering these vertices. In this section, we shall give a procedure to cover the vertices in V_{bad}, either by constructing new trees that have large size but low cost, or by merging them in to the existing λ-good trees. In the following procedure, we shall be making use of the 3-approximation algorithm by Khani and Salavatipour [16] for the MMTC problem. Recall the main theorem from above paper, re-phrased to suit our notations.

▶ **Theorem 10** ([16]). *Given an undirected graph $G(V, E)$, an integer k and edge lengths ℓ, if there exists a set of k trees (subgraphs of G) $T_1, T_2, \cdots T_k$, along with an assignment of vertices to the trees, such that $\max_{i \in [k]} \ell(T_i) \leq T^\star$ and $\cup_{i=1}^{k} \text{cov}(T_i) = V$, then there exists a polynomial time algorithm that outputs trees (subgraphs of G) $T'_1, T'_2, \cdots T'_{k'}$, along with an assignment of vertices to the trees such that $\max_{j \in [k']} \ell(T'_j) \leq 3T^\star$, $\cup_{j=1}^{k'} \text{cov}(T'_j) = V$ and $k' \leq k$.*

We refer to the algorithm referred to in the above theorem as the KS-Algorithm in subsequent sections. As mentioned earlier, although the trees T'_j might share vertices and edges, the set of vertices that such trees *cover* forms a partition of V. Note that KS algorithm takes an edge weighted graph and number of trees as input.

A Bipartite Matching Instance. Our algorithm first constructs a maximum matching instance on a suitably defined bipartite graph $H = (B, Q, F)$. In the course of algorithm, Q remains fixed. However, the other partitite set B and consequently the edge set F is refined iteratively, each time giving rise to a new maximum matching instance. Our invariants will ensure that, either we end up with a perfect matching of B or conclude that there are significantly many light trees in the optimal solution, which need to be handled separately. For clarity of notation, we parameterize set B by iteration index t - define B_t to be the partite set B at iteration t. A vertex in the set B_t represents a subtree $T_{j,t}, j = 1, 2, \cdots |B_t|$ such that $\cup_{j=1}^{|B_t|} \text{cov}(T_{j,t}) = V_{bad}$. The set Q contains a vertex corresponding to each tree in $T \in \lambda_{good}$. Next, we define the edges in F_t at any iteration t. There exists an edge

$e = (T_{j,t}, T) \in F_t, T \in Q$ if and only if $d(V(T_{j,t}), V(T)) \leq T^\star$. For a fixed t, define $\mathcal{M}_t$ to be a maximum matching instance on the graph (B_t, Q, F_t). For any subset $B' \subseteq B_t$, let $\mathcal{N}(B')$ be the neighborhood set of B' in Q.

Algorithm IterRefine. We initialize by setting $B_0 = V_{bad}$ - each vertex in B_0 is a trivial tree containing a single vertex. Now we have the required setup to describe the iterative refinement procedure. At any iteration t, we solve the maximum matching instance $\mathcal{M}_t$. If $\mathcal{M}_t$ admits a perfect matching of B_t, then we terminate. Otherwise we consider the Hall Set $S \subseteq B_t$ with maximum deficiency, that is $\text{argmax}_{S \subseteq B_t}(|S| - |\mathcal{N}(S)|)$. Let set S contains the subtrees $T_{1,t}, T_{2,t}, \cdots T_{s,t}, s = |S|$. Let $G_t = (V_t, E_t)$ be defined as: $V_t = \cup_{j=1}^{s} \text{cov}(T_{j,t}), E_t = \{(u,v) | u, v \in V_t\}$ and $\ell(u,v)$ is the length of shortest path between u and v in G. We run the KS algorithm on G_t with $k = s - 1$. Suppose it returns trees $T'_{1,t}, T'_{2,t}, \cdots T'_{s',t}$, $s' \leq s - 1$. If cost of each such tree is at most $6T^*$, we define $B_{t+1} = (B_t \setminus S) \bigcup_{j=1}^{s'} \{T'_{j,t}\}$, the edge set F_{t+1} accordingly and proceed to the next iteration, else we terminate.

Algorithm 2 Algorithm IterRefine.

Input : λ_{good}, V_{bad}
Output: A maximum matching in H_t and Unmatched Set $\overline{B_m}$
Initialize the bipartite graph $H_0 = (B_0, Q, F_0), t \leftarrow 0$
While there exists a Hall Set in B_t
Pick the Hall Set $S = \text{argmax}_{S \subseteq B_t}(|S| - |\mathcal{N}(S)|)$
Apply KS-algorithm on G_t with $k = s - 1$. Let $T'_{1,t} \cdots T'_{s',t}$ be the output of the algorithm with $s' \leq s - 1$.
If $\max_{i=1}^{s'} \ell(T'_{i,t}) > 6T^*$ **Then** STOP
Else { $B_{t+1} \leftarrow (B_t \setminus S) \bigcup_{j=1}^{s'} \{T'_{j,t}\}$ and $t \leftarrow t + 1$ }
Return Maximum Matching $\mathcal{M}_t$ and unmatched subset $\overline{B_m} \subseteq B_t$

The following lemmas are crucial to prove the correctness of our algorithm. Lemma 11 is a well known result, but we give the proof for completion. We define the projection of the optimal solution on S as follows:

$$\text{OPT}\big|_S = \{T' \in \text{OPT} : \exists T_{i,t} \in S, \text{cov}(T') \cap \text{cov}(T_{i,t}) \neq \phi\}$$

▶ **Lemma 11.** *At any iteration t, let the number of trees in $\text{OPT}\big|_S$ be m. Then there exist m trees $T_i, i \in [m]$ such that $\cup_{i=1}^{m} V(T_i) = V_t, \ell(T_i) \leq 2T^*$ and T_i is a subtree of G_t for $i \in [m]$.*

Proof. Let $T'_i, i \in [m]$ be the trees in $\text{OPT}\big|_S$. Fix a tree, say T'_i. Consider the graph with each edge of T'_i doubled. It is connected and all degrees are even, hence there exists an eulerian walk, say $v_1, v_2, \ldots, v_l$. Note that vertices in this walk may be repeated. Identify a unique vertex for each $v \in V(T'_i) \cap V_t$ in this sequence. This defines a subsequence of the walk, say $v_{i_1}, v_{i_2}, \ldots, v_{i_l}$ with every vertex in $V_t \cap V(T'_i)$ occurring exactly once. This sequence defines a path in G_t since G_t is a complete graph. The total length of path is at most $2T^*$, as length of any edge in G_t is the shortest path distance between the end points in G and total length of the walk is at most $2T^*$. Repeating this argument for all trees in $\text{OPT}\big|_S$, we get that vertices in V_t can be covered by m trees of length at most $2T^*$ each. ◀

▶ **Lemma 12.** *Suppose S is the maximum deficiency Hall Set, if any, at termination of Algorithm IterRefine. Further, let k_ℓ be the number of light trees in $\text{OPT}\big|_S$. Then $|S| - |N(S)| \leq k_\ell$.*

Proof. Let k_h be the number of heavy trees in $\text{OPT}\big|_S$. We first claim that $|S| \leq k_h + k_\ell$. Assume otherwise. But then, by Lemma 11, there exists a partition of S into exactly $k_h + k_\ell$ trees in G_t with cost at most $2T^*$, by Theorem 10, the KS-algorithm applied on the Hall Set S with $k = |S| - 1$ must have produced trees with cost at most $6T^*$ and the algorithm would not have terminated.

Now, we prove that $|N(S)| \geq k_h$. Consider any heavy tree in $\text{OPT}\big|_S$, say $\widehat{T}$. By Lemma 9, at least half of the vertices in $\widehat{T}$ are covered by λ-good trees. By construction of the bipartite graph (B_t, Q), all such λ-good trees belong to the set $N(S)$, since any two vertices in the same tree of the optimal solution can be at the most $T^\star$ distance away from each other. Thus, adding up over all heavy trees in $\text{OPT}\big|_S$, the total number of vertices covered by the λ-good trees in $N(S)$ is at least $k_h \cdot \lceil \lambda/4 \rceil$. But, each λ-good tree in $N(S)$ covers exactly $\lceil \lambda/4 \rceil$ vertices. Hence, by pigeon-hole principle, $|N(S)| \geq k_h$. Combining the above two facts, we have $|S| - |N(S)| \leq k_\ell$. ◀

▶ **Corollary 13.** *Let t' be the final iteration of the Algorithm IterRefine. Then, the total number of unmatched trees in $B_{t'}$ is at most k_{light}.*

Proof. We use the fact that the size of the maximum matching in $B_{t'}$ is exactly equal to $|B_{t'}| - (|S| - |N(S)|)$, where S is the maximum deficiency Hall Set [9]. Hence, the total number of unmatched trees in $B_{t'}$ is at exactly $|S| - |N(S)| \leq k_\ell \leq k_{light}$, by Lemma 12. ◀

We proceed to the final phase of our algorithm, equipped with the above Lemmas. Recall that, there are two types of partitions in $B_{t'}$ - either matched or unmatched by $\mathcal{M}_{t'}$. First, let us consider the matched partitions - call this set B_m. For each tree T in B_m, we take its union with the λ-good tree in Q, say $\widehat{T}$ that it is matched to in $\mathcal{M}_{t'}$. More formally, we build a tree T' by connecting $T, \widehat{T}$ using the shortest path between the closest pair of vertices in T and $\widehat{T}$. We set $\text{cov}(T') = \text{cov}(T) \cup \text{cov}(\widehat{T})$. We remove $\widehat{T}$ and add the newly constructed tree T' to the set λ-good.

Finally, we consider the set of unmatched trees $\overline{B_m}$. For each such tree, we add $\lceil \lambda/2 \rceil$ dummy vertices co-located with any arbitrary vertex in the tree. We add this tree to λ_{good}. This completes the description of the algorithm.

▶ **Lemma 14.** *At termination, Algorithm $\mathcal{B}$ returns a set of trees λ_{good} that*

1. *covers all vertices in the set $V \cup V_{dummy}$.*
2. *The cost of any tree in λ_{good} is at most $11T^\star$.*
3. *$|V \cup V_{dummy}| \leq k\lambda$*

Proof. First we show that all trees in λ_{good} are indeed λ-good, that is, they cover at least $\lambda/4$ vertices. Recall that, after algorithm $\mathcal{A}$, the set Q contains only λ-good trees, by Lemma 8. At termination of algorithm $\mathcal{B}$, each tree is either formed by joining of a tree in the set Q with some tree in B_m. Hence, the tree continues to be λ-good. The only other type of tree in λ_{good} are formed by adding $\lceil \lambda/2 \rceil$ dummy vertices to an unmatched tree in $\overline{B_m}$. Hence, they are λ-good by construction. Property 1 follows from Lemma 8 and the fact that cover of trees in $B_{t'}$ is a partition of the vertices in V_{bad}. Next we prove Property 2. There are two types of trees in λ_{good}. Let us consider any tree that is formed by the union of some matched tree in B_m, say $\mathcal{T}$ and some already existing λ-good tree in Q, say $\widehat{T}$. By Lemma 8, $\ell(\widehat{T}) \leq 4T^\star$. By Lemma 11, $\ell(\mathcal{T}) \leq 6T^\star$. A tree is formed by joining these two trees by a path of cost at the most $T^\star$. So, the resultant tree has cost at the most $(4+6+1)T^\star = 11T^\star$. The second kind of tree corresponds to the unmatched vertices in $B_{t'}$ and hence their cost is bounded above by $3T^\star$, by Lemma 11. Addition of dummy vertices does not affect the cost at all.

Property 3 can be proved as follows. Recall that $|V| \leq k_{heavy}\lambda + k_{light}\lambda/2$. Now, we add $\lambda/2$ dummy vertices to each unmatched partition in $B_{t'}$, which, by Corollary 13, can be at the most $k_{light}\lambda/2$. Hence, $|V \cup V_{dummy}| \leq k_{heavy} \cdot \lambda + 2 \cdot k_{light} \cdot \lambda/2 \leq k\lambda$ ◀

The above lemma guarantees that the total number of vertices in the graph is at the most $k\lambda$. For technical reasons that would be clear in Section 2.3, we add some more dummy vertices co-located with an arbitrarily selected $v \in V$, such that the total number of vertices is an integer multiple of λ. We note at this point that we do not make any claim about the number of trees in the set λ_{good}. In fact, the number can be larger that k, the original budget. However, in the next Section, we show how to refine the trees such that number of trees is at most k, without increasing the cost by more than a constant factor and maintaining the capacity constraints. Also note that all our procedures are polynomial time.

2.3 Algorithm $\mathcal{C}$: Converting λ-good trees to feasible trees

In the previous section, we gave algorithms that ensure the property that each tree covers at least $\lambda/4$ vertices and the cost of each tree is bounded by $\mathcal{O}(T^\star)$. In this section, we prove that it is possible to construct a solution that creates at most k admissible trees given the previous solution, where k is the number of trees that the optimal solution uses to cover the vertices in the connected component $G = (V, E)$. We prove the following theorem in this section.

▶ **Theorem 15.** *Let $T_1, T_2, \ldots, T_m$ be trees returned by the Algorithms $\mathcal{A}$ and $\mathcal{B}$ such that $|\text{cov}(T_i)| \geq \lambda/c$, where $c \geq 1$, $\ell(T_i) \leq \alpha T^*$ for $1 \leq i \leq m$. If λ divides $|\cup_{i=1}^{m} \text{cov}(T_i)|$, then there exist trees $T'_1, T'_2, \ldots, T'_l$ spanning V such that $|\text{cov}(T'_i)| = \lambda, \ell(T'_i) \leq (6c+6c\alpha-2\alpha-6)T^*$ for $1 \leq i \leq l$. Further, such trees can be found in polynomial time.*

Define G_P be to be an unweighted graph with $T_1, T_2, \ldots, T_m$ as vertices. There is an edge (T_i, T_j) if $d(T_i, T_j) = d(V(T_i), V(T_j)) \leq T^*$.

▷ Claim 16. *G_P is connected. Further, the implicit cost of each unweighted edge in G_p is at most $T^\star$.*

Proof. The claim follows from the fact that we run Algorithms $\mathcal{A}$ and $\mathcal{B}$ only on connected components of $\hat{G}$ formed after removing all edges that are of length more than $T^\star$ from G. ◁

We will need the following theorem by Karaganis [15] to prove our result:

▶ **Theorem 17** ([15])**.** *Let $G = (V, E)$ be a connected simple unweighted graph. Define G^3 to be the graph on the same set of vertices and set of edges*

$$E^3 = \{(u, v) | \textit{shortest path distance between } u \textit{ and } v \textit{ in } G \textit{ is at the most } 3\}$$

Then, G^3 has a Hamiltonian path.

By the above theorem, there exists a Hamiltonian path in G_P^3. After renaming, let $T_1, T_2, \ldots, T_m$ be the order in which the vertices of G appear on that path. Then $d(T_i, T_{i+1}) \leq 3T^* + 2(\text{max diameter of any } T_i) \leq (2\alpha + 3)T^*$ for $1 \leq i \leq m-1$. We are now ready to prove the main results of this section.

▶ **Lemma 18.** *Let $T_1, T_2, \ldots, T_m$ be trees such that $|\text{cov}(T_i)| \geq \lambda/c$ for some $c > 1$, $\ell(T_i) \leq \alpha T^*$ for $1 \leq i \leq m$ and $d(T_i, T_{i+1}) \leq D$ for $1 \leq i \leq m-1$. There exist trees $T'_1, T'_2, \ldots, T'_l$ such that $d(T'_i, T'_{i+1}) \leq D$, $|\text{cov}(T'_i)| \geq \lambda$ for $1 \leq i \leq l-1$ and $\ell((T'_i) \leq (c-1)D + c\alpha T^*$ for $1 \leq i \leq l$. Further, such trees can be computed in polynomial time.*

Proof. We make $l = \lceil m/c \rceil$ trees. Let $P(T_p, T_q)$ denote the shortest path connecting trees T_p and T_q. $T'_i = \bigcup_{j=(i-1)c+1}^{min(ic,m)} (T_j \cup P(T_j, T_{j+1}))$ for $1 \leq i \leq l$. Since $T'_1, T'_2, \ldots, T'_{l-1}$ are formed by combining c trees, each covering at least λ/c vertices, $|\text{cov}(T'_i)| \geq \lambda$ vertices. Since any V' is formed by combining at most c trees, $\ell(T') \leq (c-1)D + c\alpha T^*$. Also, by construction $d(T'_i, T'_{i+1}) \leq D$ for $1 \leq i \leq l-1$ ◀

▶ **Lemma 19.** *Let $T_1, T_2, \ldots, T_m$ be trees such that $|\text{cov}(T_i)| \geq \lambda$ for $2 \leq i \leq m$, $\ell(T_i) \leq \alpha T^*$ for $1 \leq i \leq m$ and $d(T_i, T_{i+1}) \leq D$ for $1 \leq i \leq m-1$. If λ divides $|\cup_{i=1}^{m} \text{cov}(T_i)|$, then there exist trees $T'_1, T'_2, \ldots, T'_l$ such that $|\text{cov}(T'_i)| = \lambda, \ell(T'_i) \leq D + 2\alpha T^*$ for $1 \leq i \leq l$. Further, such trees can be found in polynomial time.*

Proof. We will prove this by induction on m. If $m = 1$, then λ divides $|\text{cov}(T_1)|$. We partition $\text{cov}(T_1)$ into sets of size λ arbitrarily. Cost of a tree covering each of these partitions will be at most αT^* (T_1 is one such tree). Let $m > 1$ and $|\text{cov}(T_1)| = c\lambda + p$, where $c \geq 0, 1 \leq p \leq \lambda - 1$. We first make c partitions of size λ each out of $\text{cov}(T_1)$. Cost of spanning tree covering each of these partitions is at most αT^*. We create a new tree, say T_p, by taking p remaining points of $\text{cov}(T_1)$ and any $\lambda - p$ points of $\text{cov}(T_2)$. $|\text{cov}(T_p)| = \lambda$ and $\ell(T_p) \leq \ell(T_1) + D + \ell(T_2) \leq D + 2\alpha T^*$. The remaining points form a smaller instance and by induction can be covered by trees with desired properties. This completes the induction step and proof of the theorem. ◀

Proof of Theorem 15. From the discussion above and applying Lemma 18, we get trees covering at least λ vertices and cost at most $C = (c-1)D + c\alpha T^*$. Using Lemma 19 on these trees, we get trees covering exactly λ vertices and cost at most $D + 2C$. Plugging in $D = (2\alpha + 3)T^*$ gives the desired result. ◀

Proof of Theorem 5. Plugging in $c = 4$ and $\alpha = 11$ from Lemma 14 gives a 260-factor approximation algorithm for the capacitated min max tree cover problem. ◀

3 Extension to Capacitated Rooted Tree Cover

In this section, we show how to extend Theorem 5 to the rooted setting. Recall that in CapRMMTC, in addition to the settings of CapMMTC, we given a set of root vertices, $R = \{r_1, r_2, \cdots r_k\}$. The k output trees need to be rooted at each of these root vertices. We prove the following theorem.

▶ **Theorem 20.** *Given a graph $G = (V, E)$ with the edge lengths $\ell : E \to \mathbb{R}_{\geq 0}$, a set of roots $R \subseteq V, |R| = k$ and a parameter λ, suppose there exist subtrees of G, $T_1, T_2, \cdots, T_{k'}$, where $\cup_{i=1}^{k} \text{cov}(T_i) = V$, $\max_{i \in [k']} |\text{cov}(T_i)| \leq \lambda, V(T_i) \cap R \neq \phi$ and $\ell(T_i) \leq T^\star, \forall i = 1, 2, \cdots k$. Then there exists a polynomial time algorithm that finds subtrees of G, $T'_1, T'_2, \cdots T'_{k'}, k' \leq k$ such that $\cup_{i=1}^{k'} \text{cov}(T'_i) = V$, $\max_{i \in [k']} |\text{cov}(T'_i)| \leq \lambda$, $V(T'_i) \cap R \neq \phi$ and $\ell(T'_i) \leq \beta T^\star$, where $\beta = \mathcal{O}(1)$.*

Proof. Similar to Section 2, we focus on one of the connected components, say G' obtained after deleting all edges of length larger than $T^\star$. Let $R' = \{r_1, r_2, \cdots, r_{k'}\}$ be the subset of roots belonging to this component. There exists a partition of $V' - R'$ into k' trees, each covering at most $\lambda - 1$ vertices and cost no more than $T^\star$. Using Theorem 5, we first compute subtrees of G' , $T_1, \ldots, T_{k'}$, such that $|\text{cov}(T'_i)| \leq \lambda - 1$ and cost is $\mathcal{O}(T^\star)$. We construct a bipartite graph $H' = (\mathcal{T} \cup R', E')$, where $\mathcal{T}$ contains a vertex corresponding to each tree $T'_j, j = 1, 2, \cdots, k'$. We shall abuse notations slightly and interchangeably use T'_j to denote the vertex in $\mathcal{T}$ corresponding to the tree T'_j and the actual tree itself, noting the difference

wherever necessary. $(r_i, T'_j) \in E'$ if and only if $d(r_i, T'_j) \leq T^\star$. If there exists a matching in H' such that all vertices of $\mathcal{T}$ are matched, we add the matching root vertex to T'_j. This increases the cost of a component by at most $T^\star$. Now each component contains at least one root vertex, covers at most λ vertices and costs $\mathcal{O}(T^\star)$. If all the vertices of $\mathcal{T}$ cannot be matched, there exists a set $S \subseteq T$ such that $|N(S)| < |S|$. Let $V(S) = \cup_{T'_j \in S} V(T'_j)$. In optimal solution, every vertex of $V(S)$ is contained in a tree with one of the vertices in $N(S)$ as its root. Hence, there exists $|N(S)|$ trees such that each tree covers at most $\lambda - 1$ points of $V(S)$, costs no more than $4T^\star$ and union of all the trees cover $V(S)$. We use the result of Theorem 5 to get at most $|N(S)|$ trees covering $V(S)$, such that each tree covers at most $\lambda - 1$ vertices and costs $\mathcal{O}(T^\star)$. Note that this procedure is similar to **IterRefine** defined earlier. We now have a new set of trees. We modify our bipartite graph H' by removing vertices corresponding to trees in S and replacing them by vertices corresponding to newly formed trees. We modify E' suitably. If there exists a matching in the new graph, we are done, otherwise we repeat the above procedure till we find a perfect matching. Note that the size (in terms of number of vertices) of H' decreases by at least $|S| - |N(S)| \geq 1$ after every step and hence will halt in at most k' steps. ◀

4 Conclusion

In this paper, we give the first constant factor approximation algorithms for the hard uniform capacitated versions of unrooted and rooted Min Max Tree Cover problem on a weighted graph. Our main technical contribution lies is devising an iterative refinement procedure that distributes vertices as evenly as possible among the different trees in the cover, without incurring too much cost.

Our current approximation factor is large (> 250) and although we did not attempt to optimize, it is unlikely that our approach can reduce the factor to a very small constant. Such a result requires new ideas. Perhaps somewhat surprisingly, no LPs with constant integrality gap is known, even for the uncapacitated problem. We leave open the question of finding such an LP, in particular, configuration-style LPs, which can possibly lead to a small approximation ratio. On the other hand, the only known hardness result, to the best of our knowledge, follows from the hardness of the uncapacitated version [20]. It would be interesting to prove a better lower bound for the capacitated case, if possible.

As mentioned in Section 1, there is a PTAS known for the uncapacitated rooted MMTC problem on a tree metric. We conclude with our second open question of proving a PTAS for the hard capacitated version of the same.

References

1 Noga Alon, Yossi Azar, Gerhard J Woeginger, and Tal Yadid. Approximation schemes for scheduling on parallel machines. *Journal of Scheduling*, 1(1):55–66, 1998.

2 Hyung-Chan An, Aditya Bhaskara, Chandra Chekuri, Shalmoli Gupta, Vivek Madan, and Ola Svensson. Centrality of trees for capacitated $$k$$k-center. *Math. Program.*, 154(1-2):29–53, December 2015. doi:10.1007/s10107-014-0857-y.

3 Esther M. Arkin, Refael Hassin, and Asaf Levin. Approximations for minimum and min-max vehicle routing problems. *J. Algorithms*, 59(1):1–18, 2006. doi:10.1016/j.jalgor.2005.01.007.

4 Amariah Becker, Philip N. Klein, and Aaron Schild. A PTAS for bounded-capacity vehicle routing in planar graphs. *CoRR*, abs/1901.07032, 2019. arXiv:1901.07032.

5 Amariah Becker and Alice Paul. A PTAS for minimum makespan vehicle routing in trees. *CoRR*, abs/1807.04308, 2018. arXiv:1807.04308.

6 Jaroslaw Byrka, Bartosz Rybicki, and Sumedha Uniyal. An approximation algorithm for uniform capacitated k-median problem with 1+\epsilon capacity violation. In *Integer Programming and Combinatorial Optimization - 18th International Conference, IPCO 2016, Liège, Belgium, June 1-3, 2016, Proceedings*, pages 262–274, 2016. doi:10.1007/978-3-319-33461-5_22.

7 Lin Chen, Klaus Jansen, Wenchang Luo, and Guochuan Zhang. An efficient PTAS for parallel machine scheduling with capacity constraints. In *Combinatorial Optimization and Applications - 10th International Conference, COCOA 2016, Hong Kong, China, December 16-18, 2016, Proceedings*, pages 608–623, 2016. doi:10.1007/978-3-319-48749-6_44.

8 Lin Chen and Dániel Marx. Covering a tree with rooted subtrees - parameterized and approximation algorithms. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2801–2820, 2018. doi:10.1137/1.9781611975031.178.

9 Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.

10 Guy Even, Naveen Garg, Jochen Könemann, R. Ravi, and Amitabh Sinha. Min–max tree covers of graphs. *Operations Research Letters*, 32(4):309–315, 2004. doi:10.1016/j.orl.2003.11.010.

11 Naveen Garg. Saving an epsilon: A 2-approximation for the k-mst problem in graphs. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, STOC '05, pages 396–402, New York, NY, USA, 2005. ACM. doi:10.1145/1060590.1060650.

12 Nili Guttmann-Beck and Refael Hassin. Approximation algorithms for min–max tree partition. *Journal of Algorithms*, 24(2):266–286, 1997. doi:10.1006/jagm.1996.0848.

13 Mordecai Haimovich and Alexander HG Rinnooy Kan. Bounds and heuristics for capacitated routing problems. *Mathematics of operations Research*, 10(4):527–542, 1985.

14 Mong-Jen Kao. Iterative partial rounding for vertex cover with hard capacities. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2638–2653. SIAM, 2017.

15 Jerome J Karaganis. On the cube of a graph. *Canadian Mathematical Bulletin*, 11(2):295–296, 1968.

16 M. Reza Khani and Mohammad R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69(2):443–460, June 2014. doi:10.1007/s00453-012-9740-5.

17 Hiroshi Nagamochi and Kohei Okada. Approximating the minmax rooted-tree cover in a tree. *Information Processing Letters*, 104(5):173–178, 2007. doi:10.1016/j.ipl.2007.06.012.

18 Barna Saha and Aravind Srinivasan. A new approximation technique for resource-allocation problems. *Random Struct. Algorithms*, 52(4):680–715, 2018. doi:10.1002/rsa.20756.

19 Sam Chiu-wai Wong. Tight algorithms for vertex cover with hard capacities on multigraphs and hypergraphs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2626–2637. Society for Industrial and Applied Mathematics, 2017.

20 Zhou Xu and Qi Wen. Approximation hardness of min-max tree covers. *Oper. Res. Lett.*, 38(3):169–173, May 2010. doi:10.1016/j.orl.2010.02.004.

An Extension of Plücker Relations with Applications to Subdeterminant Maximization

Nima Anari
Department of Computer Science, Stanford University, CA, USA
anari@cs.stanford.edu

Thuy-Duong Vuong
Department of Computer Science, Stanford University, CA, USA
tdvuong@stanford.edu

Abstract

Given a matrix A and $k \geq 0$, we study the problem of finding the $k \times k$ submatrix of A with the maximum determinant in absolute value. This problem is motivated by the question of computing the determinant-based lower bound of Lovász et al. [11] on hereditary discrepancy, which was later shown to be an approximate upper bound as well [14]. The special case where k coincides with one of the dimensions of A has been extensively studied. Nikolov [17] gave a $2^{O(k)}$-approximation algorithm for this special case, matching known lower bounds; he also raised as an open problem the question of designing approximation algorithms for the general case.

We make progress towards answering this question by giving the first efficient approximation algorithm for general $k \times k$ subdeterminant maximization with an approximation ratio that depends only on k. Our algorithm finds a $k^{O(k)}$-approximate solution by performing a simple local search. Our main technical contribution, enabling the analysis of the approximation ratio, is an extension of Plücker relations for the Grassmannian, which may be of independent interest; Plücker relations are quadratic polynomial equations involving the set of $k \times k$ subdeterminants of a $k \times n$ matrix. We find an extension of these relations to $k \times k$ subdeterminants of general $m \times n$ matrices.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Randomness, geometry and discrete structures; Theory of computation → Randomized local search

Keywords and phrases Plücker relations, determinant maximization, local search, exchange property, discrete concavity, discrepancy

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.56

Category APPROX

Related Version A full version of the paper is available at `https://arxiv.org/abs/2004.13018`.

Acknowledgements We would like to thank Aleksandar Nikolov for initial discussions about general subdeterminant maximization.

1 Introduction

We consider the problem of finding the $k \times k$ submatrix of a given $m \times n$ matrix A that has the largest determinant in absolute value:

$$\operatorname{maxdet}_k(A) := \max\left\{ |\det(A_{I,J})| \;\middle|\; I \in \binom{[m]}{k}, J \in \binom{[n]}{k} \right\}.$$

A well-studied special case of this problem asks to find the maximum absolute determinant of a *maximal* submatrix. In other words, k is set to $\min\{m, n\}$. This special case is known in the literature as the largest volume simplex problem or simply (sub)determinant maximization [10, 4, 17], and it was originally framed as the problem of finding a largest simplex in


© Nima Anari and Thuy-Duong Vuong;
licensed under Creative Commons License CC-BY



a convex body, a simplex-based analog of the John ellipsoid. The best approximation algorithm for when $k = \min\{m, n\}$ was obtained by Nikolov [17] who gave an efficient $2^{O(k)}$-approximation algorithm, improving upon $\log(k)^{O(k)}$-approximation of [4], and the earlier $k^{O(k)}$-approximation of [10], and also matching known lower bounds [4].

More recently, a line of work has studied various generalizations of the largest volume simplex problem, where the returned indices of the submatrix are required to satisfy a matroid constraint [18, 2, 22, 6, 3, 12]. This line of work led to fruitful applications in several problems in combinatorial optimization: experimental design, network design, fair allocation, column subset selection, and more (see [12] for the history and applications).

Despite the extensive study of variants of the special case $k = \min\{m, n\}$, little has been done for the general case where $k < \min\{m, n\}$. A key motivation behind studying the general case comes from discrepancy theory, namely the problem of computing the determinant lower bound on hereditary discrepancy, due to Lovász et al. [11]. This quantity is defined formally as

$$\operatorname{detlb}(A) := \max\left\{ \sqrt[k]{\operatorname{maxdet}_k(A)} \;\middle|\; k \geq 0 \right\}.$$

Matoušek [14] showed, by completing earlier results of Lovász et al. [11], that $\operatorname{detlb}(A)$ is a polylogarithmic approximation to the hereditary discrepancy of A. This raised the question of efficiently approximating $\operatorname{detlb}(A)$. Nikolov and Talwar [19] showed how to approximately compute the hereditary discrepancy by bypassing $\operatorname{detlb}(A)$ and instead computing a quantity known as $\gamma_2(A)$; they showed that $\gamma_2(A)$ is a logarithmic approximation of $\operatorname{detlb}(A)$ [15] and a polylogarithmic approximation of hereditary discrepancy. But efficient $O(1)$-approximation of detlb remains open. Nikolov [17] who obtained the best approximation algorithm for the largest volume simplex problem, posed this as an open problem. Such a result has the potential to improve the approximation factor for hereditary discrepancy, as the worst known gap between detlb and hereditary discrepancy is only logarithmic [21, 14].

As a step towards answering this question, we show how to approximate $\operatorname{maxdet}_k(A)$ efficiently, with an approximation factor that depends only on k.

▶ **Theorem 1.** *There is a polynomial time algorithm that on input $A \in \mathbb{R}^{m \times n}$, outputs sets of indices $I \in \binom{[m]}{k}$ and $J \in \binom{[n]}{k}$ guaranteeing*

$$k^{O(k)} \cdot |\det(A_{I,J})| \geq \operatorname{maxdet}_k(A).$$

To the best of our knowledge, this is the first nontrivial approximation algorithm for maxdet_k. Our algorithm is based on a simple local search procedure, where in each iteration indices of *up to two* rows and/or columns are replaced by new ones, until an approximate local maximum is found.

Local search and greedy algorithms have been studied for the related problems of largest volume simplex, D-optimal design, and maximum a posteriori inference in (constrained) determinantal point processes [7, 9, 13, 8]. A key difference in our work, compared to prior works, is that we need to allow *two changes* per iteration. It is easy to construct examples where replacing only one row or one column at a time can get us stuck in an arbitrarily bad local optimum. For example, consider a diagonal matrix:

$$A := \begin{bmatrix} d_1 & 0 & \dots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & d_n \end{bmatrix}$$

Any principal $k \times k$ submatrix is a local optimum. Changing any row or column results in a 0 determinant. But obviously, d_is can be planted in a way that some of the local optima become arbitrarily bad. On the other hand, allowing simultaneous change of a row and a column lets us move between various subsets of d_is, and escape the bad local optima.

1.1 Techniques

Despite the simplicity of applying local search to combinatorial optimization problems, it is often difficult to prove approximation guarantees for its performance. We take a page from the study of matroids and discrete convexity [16], and prove a quantitative exchange inequality for subdeterminants. We will formally show that if (I, J) and (I^*, J^*) are two sets of indices determining $k \times k$ submatrices, one can swap at most two elements in total between I and I^*, and J and J^*, and obtain

$$|\det(A_{I,J})| \cdot |\det(A_{I^*,J^*})| \leq k^{O(1)} \cdot |\det(A_{I\Delta dI, J\Delta dJ})| \cdot |\det(A_{I^*\Delta dI, J\Delta dJ})| \tag{1}$$

for $dI \subseteq I\Delta I^*, dJ \subseteq J\Delta J^*$ of total size $|dI| + |dJ| \in \{2, 4\}$. This can be viewed as a form of discrete log-concavity for the determinant function on submatrices, and allows us to bound the approximation ratio of a local maximum.

Exchange properties have a long history in the theory of matroids, valuated matroids, and M-concavity [16]. Besides their use in proving the performance of greedy and local search algorithms for optimization problems, they have also recently found applications in sampling problems [1].

In order to prove the exchange inequality, we find an extension of Plücker relations to $k \times k$ subdeterminants of $m \times n$ matrices. The relations are in the form of an identity expressing the l.h.s. of Equation (1) as a linear combination of the possible values, for different choices of dI, dJ on the r.h.s. Classical Plücker relations establish exactly this form of identity in the case of $k = \min\{m, n\}$, and have been known to be connected to variants of matroids and exchange properties [5], although not quantitative exchanges of the approximate multiplicative type. Our key technical contribution is the establishment of a variant of these identities when $k < \min\{m, n\}$.

Several variants of Plücker relations have been studied in the literature. For example Dress and Wenzel [5] extended the Plücker relations to Pfaffians of skew-symmetric matrices. Their extension involves submatrices of varying sizes, and does not immediately yield a relationship involving just $k \times k$ submatrices. Both our approximate exchange inequality, and our extension of Plücker relations appear to be novel and might be of independent interest.

2 Preliminaries

We use the notation $[n] = \{1, \ldots, n\}$ for integers n. We denote the family of subsets of size k from $[n]$ by $\binom{[n]}{k}$. We use $S\Delta T = (S \setminus T) \cup (T \setminus S)$ to denote the symmetric set difference between S and T. When m, n, k are clear from context, we denote by $\mathcal{I}$ the family of valid submatrix index pairs for $k \times k$ submatrices

$$\mathcal{I} := \binom{[m]}{k} \times \binom{[n]}{k}.$$

For a pair $S = (S_{\text{row}}, S_{\text{col}}) \in \mathcal{I}$, and a matrix $A \in \mathbb{R}^{m \times n}$, we denote by $A_S = A_{S_{\text{row}}, S_{\text{col}}}$ the submatrix of A with rows and columns indexed by $S_{\text{row}}, S_{\text{col}}$ respectively. We extend set operations, such as Δ to pairs of sets denoting row and column indices in the natural way.

For example for $S = (S_{\text{row}}, S_{\text{col}})$ and $U = (U_{\text{row}}, U_{\text{col}})$ we let $S\Delta U = (S_{\text{row}}\Delta U_{\text{row}}, S_{\text{col}}\Delta U_{\text{col}})$. Similarly we let $|S| = |S_{\text{row}}| + |S_{\text{col}}|$. The reader might wish to think of pairs of row and column index sets as one single set, with the caveat that row indices are distinguished from column indices.

Throughout the paper, we keep the input matrix $A \in \mathbb{R}^{m\times n}$ for subdeterminant maximization fixed. We also assume, w.l.o.g. that $m \leq n$. For $S = (S_{\text{row}}, S_{\text{col}}) \in \mathcal{I}$, we use $[S] = [S_{\text{row}}, S_{\text{col}}]$ and $[A_S] = [A_{S_{\text{row}},S_{\text{col}}}]$, interchangeably as a shorthand for $\det(A_S) = \det(A_{S_{\text{row}},S_{\text{col}}})$.

In Section 6, we use the following famous formula for determinants of rectangular matrix products.

▶ **Fact 2** (Cauchy-Binet Formula). *Let $A \in \mathbb{R}^{m\times n}$ and $B \in \mathbb{R}^{n\times m}$. Then*

$$\det(AB) = \sum_{S\in\binom{[n]}{m}} \det(A_{[m],S}) \det(B_{S,[m]}).$$

For indices $S = (S_{\text{row}}, S_{\text{col}}), T = (T_{\text{row}}, T_{\text{col}}) \in \mathcal{I}$, let

$$d(S,T) := |S\Delta T|/2 = |S_{\text{row}}\Delta T_{\text{row}}|/2 + |S_{\text{col}}\Delta T_{\text{col}}|/2$$

be the *distance* between S and T.

Armed with this distance, we can define the neighborhoods of a submatrix indexed by $S \in \mathcal{I}$:

▶ **Definition 3.** *For $r \geq 0$ let the r-neighborhood of $S \in \mathcal{I}$ be*

$$\mathcal{N}_r(S) := \{T \in \mathcal{I} \mid d(S,T) \leq r\}.$$

3 Subdeterminant Maximization via Local Search

In this section we prove our main result, Theorem 1. Our strategy is to use a simple local search that starts with a submatrix indexed by $S \in \mathcal{I}$, and myopically finds better and better solutions by searching 2-neighborhoods until no more improvement can be found.

To make sure that our algorithm terminates within polynomial time, we will only take improvements that increase the magnitude of the determinant by at least a lower multiplicative threshold; for our purposes, even a factor 2 improvement works. We will then show how to find a good *start*, needed to bound the number of local search steps, by bootstrapping with the help of a crude approximation algorithm.

We will find a locally approximately maximum solution as defined below.

▶ **Definition 4.** *For $\alpha > 0$, we say $S \in \mathcal{I}$ is an (r,α)-local maximum if*

$$|\det(A_S)| \geq \alpha|\det(A_T)|$$

for all $T \in \mathcal{N}_r(S)$.

Algorithm 1 finds this locally approximate maximum. It starts with some arbitrary solution $S_0 \in \mathcal{I}$, and iteratively finds α-factor improvements within the 2-neighborhood, until no more improvement can be found.

It is immediate to see that when Algorithm 1 terminates, the output is a (r,α)-local maximum.

▶ **Proposition 5.** *The output of Algorithm 1 is a $(2,\alpha)$-local maximum.*

Algorithm 1 α-Local Search.

Let $S \leftarrow S_0$
while *there is* $T \in \mathcal{N}_2(S)$ *such that* $\alpha|\det(A_T)| > |\det(A_S)|$ **do**
  Let $S \leftarrow T$
end
Output $S = (S_{\text{row}}, S_{\text{col}})$

The most challenging part of local search algorithms is proving that local (approximate) optimality implies global (approximate) optimality. We appeal to approximate exchange properties that we prove for $k \times k$ subdeterminants, and show the following statement in Section 4.

▶ **Lemma 6.** *Suppose that $S \in \mathcal{I}$ is a $(2, \alpha)$-local maximum. Then S is a $(k/\alpha)^{O(k)}$-approximate global optimum:*

$$(k/\alpha)^{O(k)} \cdot |\det(A_S)| \geq \text{maxdet}_k(A).$$

We prove the remaining part of Theorem 1, that with a suitable choice for S_0, Algorithm 1 runs in polynomial time.

▶ **Proposition 7.** *The number of steps taken by Algorithm 1 starting from S_0 is at most*

$$\log_{1/\alpha}\left(\frac{\text{maxdet}_k(A)}{|\det(A_{S_0})|}\right).$$

Proof. Each iteration improves $|\det(A_S)|$ by a factor of $1/\alpha$. On the other hand, this value can never exceed $\text{maxdet}_k(A)$, and it starts as $|\det(A_{S_0})|$. ◀

In Section 6, we show how to obtain a good S_0 by a crude algorithm, that appeals to known results for the case of $k = \min\{m, n\}$. We will formally show the following.

▶ **Lemma 8.** *There is a polynomial time algorithm that returns S_0 with*

$$(n+m)^{O(k)} \cdot |\det(A_{S_0})| \geq \text{maxdet}_k(A).$$

Having all the ingredients for Theorem 1, we finish its proof.

Proof of Theorem 1. We set α to be some constant below 1, say $1/2$. We first apply Lemma 8 to obtain a good starting point S_0. If $\det(A_{S_0}) = 0$, then $\text{maxdet}_k(A) = 0$, and there is nothing to be done. Otherwise, we run Algorithm 1 with $\alpha = 1/2$. The output of the algorithm, S, is a $(2, 1/2)$-local maximum, which by Lemma 6, is a $(2k)^{O(k)} = k^{O(k)}$-approximate solution.

Each iteration of Algorithm 1 clearly runs in polynomial time, since $\mathcal{N}_2(S)$ has at most $O(k^2(m+n)^2)$ elements. So we just need to bound the number of iterations. But by Lemma 8 and Proposition 7, the number of steps is at most

$$\log\Big((n+m)^{O(k)}\Big) = O(k \log(m+n)).$$ ◀

▶ Remark 9. The approximation factor of $k^{O(k)}$ is the best possible for local search, even when we consider (c, α)-local maxima for any constant number of row/column swaps $c \in \mathbb{Z}_{>0}$. This is true even for the special case of $k = \min\{m, n\}$. To see why, consider the $\text{maxdet}_k(A)$ problem on input $A \in \mathbb{R}^{k \times 2k}$ defined by the block form

$$A = \begin{bmatrix} I_k & c^{-\frac{1}{2}} H_k \end{bmatrix}$$

where $H_k \in \mathbb{R}^{k\times k}$ is the Hadamard matrix, a matrix with ± 1 entries whose columns are orthogonal to each other. Observe that $A_{[k],[k]} = I_k$ is a $(c,1)$-local maximum, since for any $(I,J) \in \mathcal{N}_c([k],[k])$, after rearranging rows and columns, we can write

$$|\det(A_{I,J})| = \left|\det \begin{bmatrix} I_{k-c} & \star \\ 0 & c^{-\frac{1}{2}}D \end{bmatrix}\right| = |\det(c^{-\frac{1}{2}}D)| \le 1,$$

where $D \in \{\pm 1\}^{c\times c}$, and $|\det(D)| \le c^{\frac{c}{2}}$ by the Hadamard inequality. However the global optimum is achieved by the Hadamard matrix part of A. Letting $J^* = \{k+1, \cdots, 2k\}$,

$$\left|\det(A_{[n],J^*})\right| = \left|\det(c^{-\frac{1}{2}}H_k)\right| = \left(\frac{k}{c}\right)^{\frac{k}{2}} \left|\det(A_{[k],[k]})\right|.$$

In other words, the local optimum is worse than the global optimum by a factor of $(k/c)^{k/2}$.

4 Approximate Exchange and Local to Global Optimality

Here we prove Lemma 6. Our main tool will be an exchange property, that we state below. First we define the notion of an r-exchange.

▶ **Definition 10.** *Let $S,T \in \mathcal{I}$ denote two submatrices. We call $U = (U_{\text{row}}, U_{\text{col}})$ an r-exchange between S and T, if $S\Delta U$ and $T\Delta U$ are still indices of $k\times k$ submatrices, $U \subseteq S\Delta T$, and $|U| = 2r$. Note that U simply represents the exchange of r pairs of rows and/or columns between S and T. We denote by $\mathcal{E}(S,T)$, the set of all 1-exchanges and 2-exchanges between S and T.*

Now we are ready to state the key ingredient for proving local to global optimality.

▶ **Theorem 11** (Exchange Property)**.** *Let $S,T \in \mathcal{I}$ be indices of two $k\times k$ submatrices, and assume that $S \neq T$. Then*

$$|\det(A_S)| \cdot |\det(A_T)| \le O(k^2) \max\{|\det(A_{S\Delta U})| \cdot |\det(A_{T\Delta U})| \mid U \in \mathcal{E}(S,T)\}.$$

Note that Theorem 11 can be thought of a form of discrete log-concavity for subdeterminants. Starting from submatrices S,T, we move to two "nearby" submatrices $S\Delta U$ and $T\Delta U$ that are closer to T and S respectively, and then we get that up to some error terms, the average log of the determinant goes up.

We will prove Theorem 11 in Section 5 by appealing to a new extension of Plücker relations, which is an identity between subdeterminants. Here we show how to leverage Theorem 11 to show global approximate optimality from local approximate optimality. Our strategy is to start from S being the locally optimal solution and T being the globally optimal solution, and to gradually move from T to S, accumulating at most a $(k/\alpha)^{O(k)}$ loss.

Proof of Lemma 6 using Theorem 11. Let $S \in \mathcal{I}$ be a $(2,\alpha)$-local maximum and let $L \in \mathcal{I}$ be the indices of a submatrix that has the highest subdeterminant in magnitude. We first prove the following claim.

▷ Claim 12. For any $T \in \mathcal{I}$, there exists $W \in \mathcal{I}$ such that $d(S,W) \le \max(0, d(S,T)-1)$ and

$$|\det(A_T)| \le O(k^2/\alpha) \cdot |\det(A_W)|.$$

Proof of Claim 12. If $T = S$ then the claim is trivially true, since we can take $W = S$. Assume $T \neq S$. By Theorem 11, there exists $U \in \mathcal{E}(S,T)$ such that

$$\begin{aligned}|\det(A_S)| \cdot |\det(A_T)| &\leq O(k^2) \cdot |\det(A_{S\Delta U})| \cdot |\det(A_{T\Delta U})| \\ &\leq O(k^2) \cdot \frac{|\det(A_S)|}{\alpha} \cdot |\det(A_{T\Delta U})|\end{aligned}$$

where the last inequality follows from the definition of $(2,\alpha)$-local maximum.

Setting $W = T\Delta U$ and dividing both sides by $|\det(A_S)|$ gives the desired inequality. ◁

Note that initially $d(S,L) \leq 2k$. We can iteratively apply Claim 12 for up to $2k$ times, and obtain $W \in \mathcal{I}$ such that $|\det(A_L)| \leq O(k^2/\alpha)^{2k}|\det(A_W)|$ and

$$d(S,W) \leq \max(0, d(S,L) - 2k) = 0.$$

The latter condition implies $S = W$, and we are done. ◀

5 An Extension of Plücker Relations

In this section, we prove Theorem 11 by proving an extension of the Plücker relations. These are identities relating the $k \times k$ subdeterminants of a matrix. Theorem 11 will be derived from applying the triangle inequality to these identities.

To give some intuition, let us demonstrate why the regular Plücker relations, imply an exchange property when $k = \min\{m,n\}$;

5.1 Regular Plücker Relations and Exchange

W.l.o.g., let us take $k = m$ and assume $n \geq m$. Given any subsets $S,T \in \binom{[n]}{m}$, the classical Plücker relation (see, e.g., [5]) states that, for any fixed $j \in T \setminus S$

$$\det(A_{[m],S}) \det(A_{[m],T}) = \sum_{i \in S \setminus T} \delta^i_j \det(A_{[m],S\Delta\{i,j\}}) \cdot \det(A_{[m],T\Delta\{i,j\}}),$$

where $\delta^i_j \in \{\pm 1\}$ is a sign determined by the indices i and j. The triangle inequality then implies the following exchange property

$$\begin{aligned}&|\det(A_{[m],S})| \cdot |\det(A_{[m],T})| \leq \\ &\qquad k \cdot \max\big\{|\det(A_{[m],S\Delta\{i,j\}})| \cdot |\det(A_{[m],T\Delta\{i,j\}})| \;\big|\; i \in S \setminus T, j \in T \setminus S\big\}.\end{aligned}$$

This is an analog of Theorem 11, but with just one exchange between S and T. As we saw before, we cannot hope for just *one exchange* in the general case of $k < \min\{m,n\}$. But we manage to prove an extended form of Plücker relations and, by appealing to the triangle inequality, prove Theorem 11.

5.2 Extended Plücker Relations

In this subsection, we state and prove a "two-dimensional" extension of Plücker relations. In trying to find this relationship, we did a bit of guesswork; we knew we were looking for an identity involving only neighbors of the submatrices S and T, to make sure we can extract an exchange inequality. By running computer algebra systems on small values of k, we discovered the correct form of the identity, and then proceeded to prove it.

Consider $S = (S_{\text{row}}, S_{\text{col}}), T = (T_{\text{row}}, T_{\text{col}}) \in \mathcal{I}$. Note that only the entries in $A_{S\cup T}$ matter and that permuting the rows and/or columns in $S \cup T$ will preserve determinants of $k \times k$ minors up to sign.

We first show a Plücker relation for the case when S and T are disjoint, i.e., $S_{\text{row}} \cap T_{\text{row}} = S_{\text{col}} \cap T_{\text{col}} = \emptyset$. W.l.o.g., we can assume that

$$S_{\text{row}} = S_{\text{col}} = \{1, \ldots, k\} \quad \text{and} \quad T_{\text{row}} = T_{\text{col}} = \{k+1, \cdots, 2k\}, \tag{2}$$

and that A has the following block form:

$$A = \left[\begin{array}{c|c} C & V \\ \hline U & D \end{array}\right].$$

Note that $A_S = C$ and $A_T = D$.

We adopt a few notations for this section.

- We use $[U_{\text{row}}, U_{\text{col}}]$ to denote $\det(A_{U_{\text{row}}, U_{\text{col}}})$.
- Matrix entries are denoted by lowercase letter. Submatrices are denoted by uppercase letter. For example, we denote entries of submatrix C by $c_{i,j}$ for $i \in S_{\text{row}}, j \in S_{\text{col}}$.
- For a set L and $i \in L$ we use $L - i$ and L^{-i} as short hand for $L \setminus \{i\}$. Let $r_L(i)$ denote the rank of i in L, i.e., the number of $i' \in L$ that are smaller than i.
- For $U_{\text{row}} \subseteq S_{\text{row}} \Delta T_{\text{row}}, U_{\text{col}} \subseteq S_{\text{col}} \Delta T_{\text{col}}$, let $\delta^U = (-1)^{\sum_{i\in U_{\text{row}}} r_*(i) + \sum_{j \in U_{\text{col}}} r_*(j)}$, where, with some abuse of notation we use r_* for both row indices and column indices, and let

$$r_*(i) = \begin{cases} r_{S_{\text{row}}}(i) \text{ if } i \in S_{\text{row}} \\ r_{T_{\text{row}}}(i) \text{ if } i \in T_{\text{row}} \end{cases}, \quad r_*(j) = \begin{cases} r_{S_{\text{col}}}(j) \text{ if } j \in S_{\text{col}} \\ r_{T_{\text{col}}}(j) \text{ if } j \in T_{\text{col}} \end{cases}.$$

▶ **Lemma 13** (Extended Plücker Relation in the Disjoint Case). *Consider $S = (S_{\text{row}}, S_{\text{col}}), T = (T_{\text{row}}, T_{\text{col}})$ as in Equation (2).*

Let $\Omega := (S, T)$. Define

$$\begin{aligned} s_1(\Omega) &= \sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} [S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}\Delta\{j,j'\}] \times [T_{\text{row}}\Delta\{i,i'\}, T_{\text{col}}\Delta\{j,j'\}] \\ s_2(\Omega) &= (-1)^k \sum_{i,i'} \delta^{\{i,i'\},\emptyset} [S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}] \times [T_{\text{row}}\Delta\{i,i'\}, T_{\text{col}}] \\ s_3(\Omega) &= \sum_{i<h,i'<h'} \delta^{\{i,i',h,h'\},\emptyset} [S_{\text{row}}\Delta\{i,h,i',h'\}, S_{\text{col}}] \times [T_{\text{row}}\Delta\{i,h,i',h'\}, T_{\text{col}}], \end{aligned} \tag{3}$$

where in above summations, $i, h \in S_{\text{row}}, i', h' \in T_{\text{row}}, j \in S_{\text{col}}, j' \in T_{\text{col}}$.

Let $s_i := s_i(\Omega)$. Then, we have the following relation

$$s_1 - 2(k-1)s_2 - 4s_3 - k^2[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] = 0 \tag{4}$$

The proof is elementary; we only use well-known identities about the determinant and perform some algebraic manipulation.

Proof of Lemma 13. Expanding $[S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}\Delta\{j,j'\}]$ along row i', we get:

$$\begin{aligned} &[S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}\Delta\{j,j'\}] = \\ &\qquad d_{i',j'}[S_{\text{row}} - i, S_{\text{col}} - j] + \sum_{\ell \in S_{\text{col}} \setminus \{j\}} (-1)^{k + r_{S_{\text{col}}^{-j}}(\ell)} u_{i',\ell}[S_{\text{row}} - i, S_{\text{col}} - j - \ell + j']. \end{aligned}$$

Expanding $[S_{\text{row}} - i, S_{\text{col}} - j - \ell + j']$ along column j', we get:

$$[S_{\text{row}} - i, S_{\text{col}} - j - \ell + j'] = \sum_{h \in S_{\text{row}} \setminus \{i\}} (-1)^{k-1+r_{S_{\text{row}}^{-i}}(h)} v_{k,j'}[S_{\text{row}} - i - h, S_{\text{col}} - j - \ell].$$

Thus

$$[S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}\Delta\{j,j'\}] =$$
$$d_{i',j'}[S_{\text{row}} - i, S_{\text{col}} - j] - \sum_{h,\ell} (-1)^{r_{S_{\text{row}}^{-i}}(h) + r_{S_{\text{col}}^{-j}}(\ell)} u_{i',\ell} v_{k,j'}[S_{\text{row}} - i - h, S_{\text{col}} - j - \ell].$$

Similarly,

$$[T_{\text{row}}\Delta\{i,i'\}, T_{\text{col}}\Delta\{j,j'\}] =$$
$$c_{i,j}[T_{\text{row}} - i', T_{\text{col}} - j'] - \sum_{h',\ell'} (-1)^{r_{T_{\text{row}}^{-i'}}(h') + r_{T_{\text{col}}^{-j'}}(\ell')} v_{i,\ell'} u_{h',j}[T_{\text{row}} - i' - h', T_{\text{col}} - j' - \ell']$$

Now, consider $[S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}\Delta\{j,j'\}] \times [T_{\text{row}}\Delta\{i,i'\}, T_{\text{col}}\Delta\{j,j'\}]$ as a multivariate polynomial p in variables $\vec{u} = \{u_{\cdot,\cdot}\}, \vec{v} = \{v_{\cdot,\cdot}\}$. For $s \in \{0,1,2\}$ let $p_s^{i,i',j,j'}$ denote the sum over monomials of p which have degree s in $\vec{u}$ and in $\vec{v}$. We will omit the superscript when appropriate.

We further decompose p_1 into $p_1 = -(p_{1A} + p_{1B})$, where

$$p_{1A} = \sum_{h,\ell} (-1)^{r_{S_{\text{row}}^{-i}}(h) + r_{S_{\text{col}}^{-j}}(\ell)} c_{i,j}[T_{\text{row}} - i', T_{\text{col}} - j'] u_{i',\ell} v_{h,j'}[S_{\text{row}} - i - h, S_{\text{col}} - j - \ell],$$

and

$$p_{1B} = \sum_{h',\ell'} (-1)^{r_{T_{\text{row}}^{-i'}}(h') + r_{T_{\text{col}}^{-j'}}(\ell')} d_{i',j'}[S_{\text{row}} - i, S_{\text{col}} - j] v_{i,\ell'} u_{h',j}[T_{\text{row}} - i' - h', T_{\text{col}} - j' - \ell'].$$

▷ Claim 14. We have

$$\sum \delta^{\{i,i'\},\{j,j'\}} p_1^{i,i',j,j'} = 2(k-1)(-1)^k \sum_{h,i'} (-1)^{h+i'} [S_{\text{row}} - h + i', S_{\text{col}}] \cdot [T_{\text{row}} - i' + h, T_{\text{col}}] \quad (5)$$

Proof. In $\sum_{i,j} \delta^{\{i,i'\},\{j,j'\}} p_{1A}^{i,i',j,j'}$ we consider the sum of all terms with the same i', j', h, l. Note that $r_{S_{\text{col}}^{-j}}(\ell) + r_{S_{\text{col}}}(j) = r_{S_{\text{col}}}(\ell) + r_{S_{\text{col}}^{-\ell}}(j) + 1 \mod 2$. This is because,

$$r_{S_{\text{col}}^{-j}}(\ell) = \begin{cases} r_{S_{\text{col}}}(\ell) \text{ if } \ell < j \\ r_{S_{\text{col}}}(\ell) - 1 \text{ if } \ell > j \end{cases} \quad \text{and } r_{S_{\text{col}}^{-\ell}}(j) = \begin{cases} r_{S_{\text{col}}}(j) - 1 \text{ if } \ell < j \\ r_{S_{\text{col}}}(j) \text{ if } \ell > j \end{cases}.$$

Similarly, $r_{S_{\text{row}}^{-i}}(h) + r_{S_{\text{row}}}(i) = r_{S_{\text{row}}}(h) + r_{S_{\text{row}}^{-h}}(i) + 1 \mod 2$. Thus, this sum is exactly,

$$\delta^{\{i',h\},\{j',\ell\}} u_{i',\ell} v_{h,j'}[T_{\text{row}} - i', T_{\text{col}} - j'] \times$$
$$\sum_{i \neq h, j \neq \ell} (-1)^{r_{S_{\text{col}}^{-\ell}}(j) + r_{S_{\text{row}}^{-h}}(i)} c_{i,j}[S_{\text{row}} - i - h, S_{\text{col}} - j - \ell] =$$
$$(k-1)\delta^{\{i',h\},\{j',\ell\}} u_{i',\ell} v_{h,j'}[T_{\text{row}} - i', T_{\text{col}} - j'][S_{\text{row}} - h, S_{\text{col}} - \ell],$$

Indeed, for each $i \in S_{\text{row}} - h$, expanding $[S_{\text{row}} - h, S_{\text{col}} - \ell]$ along row i gives

$$\sum_{j \neq \ell} (-1)^{r_{S_{\text{col}}^{-\ell}}(j) + r_{S_{\text{row}}^{-h}}(i)} c_{i,j}[S_{\text{row}} - i - h, S_{\text{col}} - j - \ell].$$

Taking sum over $i \in S_{\text{row}} - h$ gives the above equality.

Thus

$$\begin{aligned}
&\sum \delta^{\{i,i'\},\{j,j'\}} p_{1A}^{i,i',j,j'} \\
&= (k-1) \sum_{i',j',h,\ell} \delta^{\{i',h\},\{j',\ell\}} u_{i',\ell} v_{h,j'} [T_{\text{row}} - i', T_{\text{col}} - j'][S_{\text{row}} - h, S_{\text{col}} - \ell] \\
&= -(k-1)(-1)^k \sum_{h,i'} \delta^{\{i',h\},\emptyset} \Big((\sum_{\ell} (-1)^{h + r_{S_{\text{col}}}(\ell)} u_{i',\ell} [S_{\text{row}} - h, S_{\text{col}} - \ell]) \times \\
&\quad (\sum_{j'} (-1)^{1 + r_{T_{\text{col}}}(j')} v_{h,j'} [T_{\text{row}} - i', T_{\text{col}} - j']) \Big) \\
&= -(k-1)(-1)^k \sum_{h,i'} \delta^{\{i',h\},\emptyset} [S_{\text{row}} - h + i', S_{\text{col}}][T_{\text{row}} - i' + h, T_{\text{col}}]
\end{aligned}$$

Similarly,

$$\sum p_{1B} = -(k-1)(-1)^k \sum_{i,h'} \delta^{\{i',h\}} [S_{\text{row}} - i + h', S_{\text{col}}][T_{\text{row}} - h' + i, T_{\text{col}}] \qquad \lhd$$

Next, we show

▷ Claim 15.

$$\sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} p_2^{i,i',j,j'} = 4 \sum_{i<h,i'<h'} [S_{\text{row}} - i - h + i' + h', S_{\text{col}}][T_{\text{row}} - i' - h' + i + h, T_{\text{col}}] \quad (6)$$

Proof. Recall that $p_2^{i,i',j,j'}$ equals

$$\sum_{h,\ell,h',\ell'} (-1)^{\omega(h,h',\ell,\ell')} u_{i',\ell} v_{i,\ell'} v_{h,j'} u_{h',j} [S_{\text{row}} \setminus \{i,h\}, S_{\text{col}} \setminus \{j,\ell\}][T_{\text{row}} \setminus \{i',h'\}, T_{\text{col}} \setminus \{j',\ell'\}],$$

where $\omega(h,h',\ell,\ell') = r_{S_{\text{row}}^{-i}}(h) + r_{S_{\text{col}}^{-j}}(\ell) + r_{T_{\text{row}}^{-i'}}(h') + r_{T_{\text{col}}^{-j'}}(\ell')$. Taking sum and rearranging terms, we have

$$\sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} p_2^{i,i',j,j'} = \sum_{i,i',h,h'} (-1)^{r_{S_{\text{row}}}(i) + r_{T_{\text{row}}}(i') + r_{S_{\text{row}}^{-i}}(h) + r_{T_{\text{row}}^{-i'}}(h')} X_{i,i',h,h'} \times Y_{i,i',h,h'}$$

where

$$\begin{aligned}
X_{i,i',h,h'} &= \sum_{j,\ell} (-1)^{r_{S_{\text{col}}^{-j}}(\ell) + r_{S_{\text{col}}}(j)} u_{i',\ell} u_{h',j} [S_{\text{row}} - i - h, S_{\text{col}} - j - \ell] \\
Y_{i,i',h,h'} &= \sum_{j',\ell'} (-1)^{r_{T_{\text{col}}^{-j'}}(\ell') + r_{T_{\text{col}}}(j')} v_{i,\ell'} v_{h,j'} [T_{\text{row}} - i' - h', T_{\text{col}} - j' - \ell']
\end{aligned}$$

Expanding $[S_{\text{row}} - i - h + i' + h', S_{\text{col}}]$ along row h' then i', we get that $[S_{\text{row}} \Delta \{i, i', h, h'\}, S_{\text{col}}]$

$$\begin{aligned}
&= \sum_{j} (-1)^{r_{S_{\text{col}}}(j) + k + \mathbb{1}\{i' > h'\}} u_{h',j} [S_{\text{row}} - i - h + i', S_{\text{col}} - j] \\
&= \sum_{j,\ell} (-1)^{r_{S_{\text{col}}}(j) + k + \mathbb{1}\{i' > h'\}} (-1)^{r_{S_{\text{col}}^{-j}}(\ell) + k - 1} u_{i',\ell} u_{h',j} [S_{\text{row}} - i - h, S_{\text{col}} - j - \ell] \\
&= (-1)^{\mathbb{1}\{i' < h'\}} X_{i,i',h,h'}
\end{aligned}$$

Similarly, $Y_{i,i',h,h'} = (-1)^{\mathbb{1}\{i<h\}}[T_{\text{row}}\Delta\{i,i',h,h'\},T_{\text{col}}]$. Note that $\mathbb{1}\{i<h\} + r_{S_{\text{row}}}(i) + r_{S_{\text{row}}^{-i}}(h) \equiv 0 \pmod 2$. A similar equation holds for i',h'. Substituting back in we get the desired equation

$$\sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} p_2^{i,i',j,j'} = 4 \sum_{i<h,i'<h'} [S_{\text{row}}\Delta\{i,i',h,h'\},S_{\text{col}}][T_{\text{row}}\Delta\{i,i',h,h'\},T_{\text{col}}] \quad \lhd$$

Lastly, we compute $\sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} p_0^{i,j,i',j'}$. By rearranging terms and using the determinant expansion for $[S_{\text{row}},S_{\text{col}}]$ and $[T_{\text{row}},T_{\text{col}}]$, we get:

$$\begin{aligned}
&\sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} p_0^{i,j,i',j'} \\
&= \sum \delta^{\{i,i'\},\{j,j'\}} (d_{i',j'}[S_{\text{row}}-i,S_{\text{col}}-j]c_{i,j}[T_{\text{row}}-i',T_{\text{col}}-j']) \\
&= \left(\sum_{i,j} (-1)^{r_{S_{\text{row}}}(i)+r_{S_{\text{col}}}(j)} c_{i,j}[S_{\text{row}}-i,S_{\text{col}}-j]\right) \times \\
&\quad \left(\sum_{i',j'} (-1)^{r_{T_{\text{row}}}(i')+r_{T_{\text{col}}}(j')} d_{i',j'}[T_{\text{row}}-i',T_{\text{col}}-j']\right) \\
&= (t[S_{\text{row}},S_{\text{col}}])(t[T_{\text{row}},T_{\text{col}}]) \qquad (7)
\end{aligned}$$

Substituting equations Equations (5) to (7) back into s_1 we get Equation (4). ◀

Now consider the general case when $S_{\text{row}},T_{\text{row}}$ and $S_{\text{col}},T_{\text{col}}$ are not necessarily disjoint. We will create a new larger matrix A with a new set of row and column indices. In particular we create new disjoint subsets $S^*_{\text{row}},T^*_{\text{row}}$ and $S^*_{\text{col}},T^*_{\text{col}}$ with copied versions of common rows and columns. We use Lemma 13 for $S^*_{\text{row}},T^*_{\text{row}},S^*_{\text{col}},T^*_{\text{col}}$, then argue that any nonzero terms in Equation (4) must be equal to $[S_{\text{row}}\Delta U_{\text{row}},S_{\text{col}}\Delta U_{\text{col}}][T_{\text{row}}\Delta U_{\text{row}},T_{\text{col}}\Delta U_{\text{col}}]$ for some $U \subseteq S_{\text{row}}\Delta T_{\text{row}}, U_{\text{col}} \subseteq S_{\text{col}}\Delta T_{\text{col}}$.

Let $r := |S_{\text{row}} \cap T_{\text{row}}|, c := |S_{\text{col}} \cap T_{\text{col}}|$. W.l.o.g., we can assume

$$\begin{aligned}
&S_{\text{row}} = \{1,\cdots,r,r+1,\cdots,k\}, T_{\text{row}} = \{1,\cdots,r,k+(r+1),\cdots,2k\}, \\
&S_{\text{col}} = \{1,\cdots,c,c+1,\cdots,k\}, T_{\text{col}} = \{1,\cdots,c,k+(c+1),\cdots,2k\}. \qquad (8)
\end{aligned}$$

For $i \in [r]$, set row $k+i$ to be identical to row i. For $j \in [c]$, set column $k+j$ to be identical to row j.

Let $S^*_{\text{row}} := S_{\text{row}}, S^*_{\text{col}} := S_{\text{col}}, T^*_{\text{row}} = \{k+1,\cdots,2k\}, T^*_{\text{col}} = \{k+1,\cdots,2k\}$. Clearly, $S^*_{\text{row}} \cap T^*_{\text{row}} = S^*_{\text{col}} \cap T^*_{\text{col}} = \emptyset$.

Let $\Omega^* = (S^*,T^*)$ and $s^*_i := s_i(\Omega^*)$ as in Equation (3). We first prove the following claims on the structure of nonzero terms in s^*_1, s^*_2, s^*_3.

▷ Claim 16. Consider $U_{\text{row}} \subseteq S^*_{\text{row}}, U'_{\text{row}} \subseteq T^*_{\text{row}}$ of the same cardinality. Let $\overline{U}_{\text{row}} = U_{\text{row}} \cup U'_{\text{row}}$. Consider sets V,W of the same cardinality t that partition $S^*_{\text{col}} \cup T^*_{\text{col}}$.

1. If there exists $i \in U_{\text{row}} \cap [r]$ such that $k+i \notin U'_{\text{row}}$ then

$$[S^*_{\text{row}}\Delta\overline{U}_{\text{row}},V][T^*_{\text{row}}\Delta\overline{U}_{\text{row}},W] = 0.$$

2. If there exists $k+i \in U'_{\text{row}} \cap \{k+1,\cdots,k+r\}$ such that $i \notin U_{\text{row}}$ then

$$[S^*_{\text{row}}\Delta\overline{U}_{\text{row}},V][T^*_{\text{row}}\Delta\overline{U}_{\text{row}},W] = 0.$$

▷ Claim 17. Consider $U_{\text{col}} \subseteq S^*_{\text{col}}, U'_{\text{col}} \subseteq T^*_{\text{col}}$ of the same cardinality. Let $\overline{U}_{\text{col}} = U_{\text{col}} \cup U'_{\text{col}}$. Consider sets V, W of the same cardinality t that partition $S^*_{\text{row}} \cup T^*_{\text{row}}$.

1. If there exists $i \in U_{\text{col}} \cap [c]$ such that $k + i \notin U'_{\text{col}}$ then

$$[V, S^*_{\text{col}} \Delta \overline{U}_{\text{col}}][W, T^*_{\text{col}} \Delta \overline{U}_{\text{col}}] = 0.$$

2. If there exists $k + i \in U'_{\text{col}} \cap \{k+1, \cdots, k+c\}$ such that $i \notin U_{\text{col}}$ then

$$[V, S^*_{\text{col}} \Delta \overline{U}_{\text{col}}][W, T^*_{\text{col}} \Delta \overline{U}_{\text{col}}] = 0.$$

We prove Claim 16. The argument for Claim 17 is similar.

Proof of Claim 16. We prove the first statement. The second one follows by a similar argument, since the role of $U_{\text{row}}, U'_{\text{row}}$ are symmetric.

Suppose there exists $i \in U_{\text{row}} \cap [c]$ such that $k + i \notin U'_{\text{row}}$. Then $T^*_{\text{row}} \Delta \overline{U}_{\text{row}}$ contains both rows i and $k + i$, which are identical by our construction, thus $[T^*_{\text{row}} \Delta \overline{U}_{\text{row}}, W] = 0$. ◁

▶ **Lemma 18.** *Consider $S = (S_{\text{row}}, S_{\text{col}}), T = (T_{\text{row}}, T_{\text{col}})$ as in Equation (8).*
Let $\Omega := (S, T), r := |S_{\text{row}} \cap T_{\text{row}}|, c := |S_{\text{col}} \cap T_{\text{col}}|$.
Define

$$\begin{aligned}
s_1(\Omega) &= (-1)^{r+c} \sum_{i,j,i',j'} \delta^{\{i,i'\},\{j,j'\}} [S_{\text{row}} \Delta \{i,i'\}, S_{\text{col}} \Delta \{j,j'\}] \times [T_{\text{row}} \Delta \{i,i'\}, T_{\text{col}} \Delta \{j,j'\}] \\
s_2(\Omega) &= (-1)^{k-r} \sum_{i,i'} \delta^{\{i,i'\},\emptyset} [S_{\text{row}} \Delta \{i,i'\}, S_{\text{col}}][T_{\text{row}} \Delta \{i,i'\}, T_{\text{col}}] \\
\hat{s}_2(\Omega) &= (-1)^{k-c} \sum_{j,j'} \delta^{\emptyset,\{j,j'\}} [S_{\text{row}}, S_{\text{col}} \Delta \{j,j'\}][T_{\text{row}}, T_{\text{col}} \Delta \{j,j'\}] \\
s_3(\Omega) &= \sum_{i<h,i'<h'} \delta^{\{i,h,i',h'\},\emptyset} [S_{\text{row}} \Delta \{i,h,i',h'\}, S_{\text{col}}][T_{\text{row}} \Delta \{i,h,i',h'\}, T_{\text{col}}] \\
\hat{s}_3(\Omega) &= \sum_{j<\ell,j'<\ell'} \delta^{\emptyset,\{j,\ell,j',\ell'\}} [S_{\text{row}}, S_{\text{col}} \Delta \{j,\ell,j',\ell'\}][T_{\text{row}}, T_{\text{col}} \Delta \{j,\ell,j',\ell'\}],
\end{aligned} \tag{9}$$

where in above summations, $j, \ell \in S_{\text{col}} \setminus T_{\text{col}}; j', \ell' \in T_{\text{col}} \setminus S_{\text{col}}; i, h \in S_{\text{row}} \setminus T_{\text{row}}; i', h' \in T_{\text{row}} \setminus S_{\text{row}}$.

Let $s_i := s_i(\Omega)$. we have the following relations.

$$\begin{aligned}
(k^2 - 2(k-1)r + 4\binom{r}{2} - rc)[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] &= s_1 - r\hat{s}_2 - (2(k-1) + c - 4r)s_2 - 4s_3 \\
(k^2 - 2(k-1)c + 4\binom{c}{2} - rc)[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] &= s_1 - cs_2 - (2(k-1) + r - 4c)\hat{s}_2 - 4\hat{s}_3
\end{aligned} \tag{10}$$

Summing the two equations above, we get:

$$\begin{aligned}
((k-r)^2 + (k-c)^2 + (r-c)^2)[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] = \\
2s_1 - 2(k-1+r-2c)\hat{s}_2 - 2(k-1+c-2r)s_2 - 4(s_3 + \hat{s}_3)
\end{aligned} \tag{11}$$

Proof. We prove the first statement. The second one can be obtained by switching the role of columns and rows. Consider s^*_1, and let

$$X_{i,i',j,j'} := \delta^{\{i,i'\},\{j,j'\}} [S^*_{\text{row}} \Delta \{i,i'\}, S^*_{\text{col}} \Delta \{j,j'\}][T^*_{\text{row}} \Delta \{i,i'\}, T^*_{\text{col}} \Delta \{j,j'\}].$$

By Claim 16 and Claim 17, any nonzero term $X_{i,i',j,j'}$ in s_1^* must belong to one of the following cases:

1. $i \in [r], i' = k+i, j \notin [c], j' \notin \{k+1, \cdots, k+c\}$: Note that $j \in S_{\text{col}} \setminus T_{\text{col}}, j' \in T_{\text{col}} \setminus S_{\text{col}}$. Obviously $\delta^{\{i,i'\},\{j,j'\}} = \delta^{\emptyset,\{j,j'\}}$.
   Since rows $k+i$ and i are identical,

$$[S^*_{\text{row}}\Delta\{i,i'\}, S^*_{\text{col}}\Delta\{j,j'\}] = (-1)^{k+i}[S^*_{\text{row}}, S^*_{\text{col}}\Delta\{j,j'\}] = (-1)^{k+i}[S_{\text{row}}, S_{\text{col}}\Delta\{j,j'\}],$$

   and

$$[T^*_{\text{row}}\Delta\{i,i'\}, T^*_{\text{col}}\Delta\{j,j'\}] = (-1)^{1+i}[T^*_{\text{row}}, T^*_{\text{col}}\Delta\{j,j'\}] = (-1)^{1+i+c}[T_{\text{row}}, T_{\text{col}}\Delta\{j,j'\}].$$

   Therefore,

$$X_{i,i',j,j'} = (-1)^{k-c+1}\delta^{\emptyset,\{j,j'\}}[S_{\text{row}}, S_{\text{col}}\Delta\{j,j'\}][T_{\text{row}}, T_{\text{col}}\{j,j'\}].$$

2. $j \in [c], j' = k+j, i \notin [r], i' \notin \{k+1, \cdots, k+r\}$: similarly,

$$X_{i,i',j,j'} = (-1)^{k-r+1}\delta^{\{i,i'\},\emptyset}[S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}][T_{\text{row}}\Delta\{i,i'\}, T_{\text{col}}].$$

3. $i \in [r], j \in [c], i' = k+i, j' = k+j$: $X_{i,i',j,j'} = [S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}]$.
4. $i \in S_{\text{row}} \setminus T_{\text{row}}, i' \in T_{\text{row}} \setminus S_{\text{row}}, j \in j \in S_{\text{col}} \setminus T_{\text{col}}, j' \in T_{\text{col}} \setminus S_{\text{col}}$, then

$$\begin{aligned} X_{i,i',j,j'} &= \delta^{\{i,i'\},\{j,j'\}}[S^*_{\text{row}}\Delta\{i,i'\}, S^*_{\text{col}}\Delta\{j,j'\}][T^*_{\text{row}}\Delta\{i,i'\}, T^*_{\text{col}}\Delta\{j,j'\}] \\ &= (-1)^{r+c}\delta^{\{i,i'\},\{j,j'\}}[S_{\text{row}}\Delta\{i,i'\}, S_{\text{col}}\Delta\{j,j'\}][T_{\text{row}}\Delta\{i,i'\}, T_{\text{col}}\Delta\{j,j'\}] \end{aligned}$$

We can rewrite

$$s_1^* = s_1 + rc[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] - (r\,\hat{s}_2 + c\,s_2)$$

By a similar argument

$$s_2^* = s_2 - r[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}]$$
$$s_3^* = s_3 + \binom{r}{2}[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] - rs_2$$

Substitute into Equation (4) we get:

$$(k^2 - 2(k-1)r + 4\binom{r}{2} - rc)[S_{\text{row}}, S_{\text{col}}][T_{\text{row}}, T_{\text{col}}] = s_1 - r\hat{s}_2 - (2(k-1) + c - 4r)s_2 - 4s_3$$ ◀

5.3 From the Extended Plücker Relations to Exchange

Armed with the extended Plücker relations, we are now ready to prove Theorem 11.

Proof of Theorem 11. We can permute the rows and columns of A so that S_{row}, S_{col}, T_{row}, and T_{col} are as in Equation (8), while preserving the absolute value of determinant of minors. W.l.o.g., we can assume the permutation has already been applied; thus Equation (11) holds.

Let $r = |S_{\text{row}} \cap T_{\text{row}}|, c = |S_{\text{col}} \cap T_{\text{col}}|$.

With $\Omega = (S, T)$, we define

$$\begin{aligned}
\mathcal{E}_1^\Omega &:= \{(\{i, i'\}, \{j, j'\}) \mid i \in S_{\text{row}} \setminus T_{\text{row}}, i' \in T_{\text{row}} \setminus S_{\text{row}}, j \in S_{\text{col}} \setminus T_{\text{col}}, j' \in T_{\text{col}} \setminus S_{\text{col}}\} \\
\mathcal{E}_2^\Omega &:= \{(\{i, i'\}, \emptyset) \mid i \in S_{\text{row}} \setminus T_{\text{row}}, i' \in T_{\text{row}} \setminus S_{\text{row}}\} \\
\hat{\mathcal{E}}_2^\Omega &:= \{(\emptyset, \{j, j'\}) \mid j \in S_{\text{col}} \setminus T_{\text{col}}, j' \in T_{\text{col}} \setminus S_{\text{col}}\} \\
\mathcal{E}_3^\Omega &:= \{(\{i, h, i', h'\}, \emptyset) \mid i, h \in S_{\text{row}} \setminus T_{\text{row}}; i', h' \in T_{\text{row}} \setminus S_{\text{row}}\} \\
\hat{\mathcal{E}}_3^\Omega &:= \{(\emptyset, \{j, \ell, j', \ell'\}) \mid j, \ell \in S_{\text{col}} \setminus T_{\text{col}}; j', \ell' \in T_{\text{col}} \setminus S_{\text{col}}\}
\end{aligned} \tag{12}$$

then $\mathcal{E}(S, T) = \mathcal{E}_1^\Omega \cup \mathcal{E}_2^\Omega \cup \hat{\mathcal{E}}_2^\Omega \cup \mathcal{E}_3^\Omega \cup \hat{\mathcal{E}}_3^\Omega$, where $\mathcal{E}(\Omega)$ is as defined in Definition 10.

Note that

$$|\mathcal{E}_1| = (k-r)^2(k-c)^2, |\mathcal{E}_2| = (k-r)^2, |\hat{\mathcal{E}}_2| = (k-c)^2, |\mathcal{E}_3| = \binom{k-r}{2}^2, |\hat{\mathcal{E}}_3| = \binom{k-c}{2}^2.$$

Let $\gamma := \max\{|\det(A_{S\Delta U})| \cdot |\det(A_{T\Delta U})| \mid U \in \mathcal{E}(S, T)\}$. By the triangle inequality,

$$|s_1| \leq |\mathcal{E}_1|\gamma.$$

A similar inequality holds for $s_2, \hat{s}_2, s_3, \hat{s}_3$.

Consider Equation (11). Let $M := (k-r)^2 + (k-c)^2 + (r-c)^2$. By the triangle inequality and the above observation,

$$\begin{aligned}
&M \cdot [S_{\text{row}}, S_{\text{col}}] \cdot [T_{\text{row}}, T_{\text{col}}] \\
&\leq \gamma(2|\mathcal{E}_1| + |k-1+c-2r| \cdot |\mathcal{E}_2| + |k-1+r-2c| \cdot |\hat{\mathcal{E}}_2| + 4(|\mathcal{E}_3| + |\hat{\mathcal{E}}_3|)) \\
&\leq \gamma\left(((k-r)^2 + (k-c)^2)^2 + 8k((k-r)^2 + (k-c)^2)\right) \\
&\leq M(2k^2 + 8k).
\end{aligned}$$

Since $S \neq T$ so $M > 0$. Dividing both sides by M gives the desired inequality. ◀

6 A Crude Approximation Algorithm

In this section we describe a crude approximation algorithm that can be used to provide the starting point for Algorithm 1. We will formally prove Lemma 8. Our strategy is to appeal to prior results on simpler variants of determinant maximization. Specifically we use the following result of Nikolov [17]:[1]

▶ **Theorem 19** ([17]). *There is a polynomial time algorithm that given a positive semidefinite matrix $B \in \mathbb{R}^{n \times n}$ and $k \geq 0$, outputs a set $S \in \binom{[n]}{k}$ that approximately maximizes $\det(B_{S,S})$. The approximation factor of this algorithm is guaranteed to be $2^{O(k)}$.*

Using Theorem 19, we will provide an algorithm that constructs S_0, a $(n+m)^{O(k)}$-approximation to $\text{maxdet}_k(A)$ in the general case where $k < \min\{m, n\}$.

Proof of Lemma 8. Consider the following procedure that outputs $S = (S_{\text{row}}, S_{\text{col}}) \in \mathcal{I}$:

1. Let $B := AA^\intercal \in \mathbb{R}^{m \times m}$. Note that B is positive semidefinite. Use Theorem 19 to pick $S_{\text{row}} \in \binom{[m]}{k}$ that approximately maximizes $|\det(B_{S_{\text{row}}, S_{\text{row}}})|$.

[1] We remark that the approximation factor of $2^{O(k)}$ is not very important, and one can use simpler and cruder algorithms, such as [20], instead of [17].

2. Let $C := A_{S_{\text{row}},[n]} \in \mathbb{R}^{k\times n}, D := C^\intercal C \in \mathbb{R}^{n\times n}$. Use Theorem 19 to pick $S_{\text{col}} \in \binom{[n]}{k}$ that approximately maximizes $|\det(D_{S_{\text{col}},S_{\text{col}}})|$.

   We claim that for $S = (S_{\text{row}}, S_{\text{col}})$:

   $(n+m)^{O(k)} \cdot |\det(A_S)| \geq \max\{|\det(A_T)| \mid T \in \mathcal{I}\}$.

Let $T \in \mathcal{I}$ denote the indices of the submatrix with the maximum $k \times k$ subdeterminant. Note that $B_{S_{\text{row}},S_{\text{row}}} = CC^\intercal$. Thus, by the Cauchy-Binet formula,

$$\begin{aligned}\det(B_{S_{\text{row}},S_{\text{row}}}) &= \sum_{W_{\text{col}}\in\binom{[n]}{k}} \det(C_{[k],W_{\text{col}}}) \det(C^\intercal_{W_{\text{col}},[k]}) = \sum_{W_{\text{col}}\in\binom{[n]}{k}} \det(C_{[k],W_{\text{col}}})^2 \\ &= \sum_{W_{\text{col}}\in\binom{[n]}{k}} \det(A_{S_{\text{row}},W_{\text{col}}})^2 \leq \sum_{W_{\text{col}}\in\binom{[n]}{k}} 2^{O(k)} \cdot \det(A_{S_{\text{row}},S_{\text{col}}})^2 \\ &= n^{O(k)} \cdot \det(A_{S_{\text{row}},S_{\text{col}}})^2. \end{aligned} \tag{13}$$

Similarly, the Cauchy-Binet formula applied to $B_{T_{\text{row}},T_{\text{row}}} = A_{T_{\text{row}},[n]}(A_{T_{\text{row}},[n]})^\intercal$ gives

$$\det(B_{T_{\text{row}},T_{\text{row}}}) = \sum_{W_{\text{col}}\in\binom{[n]}{k}} \det(A_{T_{\text{row}},W_{\text{col}}})^2 \geq \det(A_{T_{\text{row}},T_{\text{col}}})^2 \tag{14}$$

Thus,

$$n^{O(k)} \det(A_{S_{\text{row}},S_{\text{col}}})^2 \geq 2^{O(k)} \det(B_{S_{\text{row}},S_{\text{row}}}) \geq \det(B_{T_{\text{row}},T_{\text{row}}}) \geq \det(A_{T_{\text{row}},T_{\text{col}}})^2,$$

where the first inequality follows from (13) and the second from definition of S_{row}. ◀

References

1. Nima Anari, Kuikui Liu, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials iv: Exchange properties, tight mixing times, and faster sampling of spanning trees. *arXiv preprint*, 2020. arXiv:2004.07220.
2. Nima Anari and Shayan Oveis Gharan. A generalization of permanent inequalities and applications in counting and optimization. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 384–396. ACM, June 2017. doi:10.1145/3055399.3055469.
3. Nima Anari, Shayan Oveis Gharan, and Cynthia Vinzant. Log-concave polynomials i: Entropy and a deterministic approximation algorithm for counting bases of matroids. In *Proceedings of the 59th IEEE Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, October 2018. doi:10.1109/focs.2018.00013.
4. Marco Di Summa, Friedrich Eisenbrand, Yuri Faenza, and Carsten Moldenhauer. On largest volume simplices and sub-determinants. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 315–323. SIAM, 2014.
5. Andreas Dress and Walter Wenzel. Grassmann-plücker relations and matroids with coefficients. *Advances in Mathematics*, 86(1):68–110, 1991.
6. Javad B Ebrahimi, Damian Straszak, and Nisheeth K Vishnoi. Subdeterminant maximization via nonconvex relaxations and anti-concentration. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1020–1031. Ieee, 2017.
7. Valerii Vadimovich Fedorov. *Theory of optimal experiments*. Elsevier, 2013.
8. Piotr Indyk, Sepideh Mahabadi, Shayan Oveis Gharan, and Alireza Rezaei. Composable core-sets for determinant maximization problems via spectral spanners. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1675–1694. SIAM, 2020.

9 Tarun Kathuria and Amit Deshpande. On sampling and greedy map inference of constrained determinantal point processes. *arXiv preprint*, 2016. arXiv:1607.01551.

10 Leonid Khachiyan. On the complexity of approximating extremal determinants in matrices. *Journal of Complexity*, 11(1):138–153, 1995.

11 László Lovász, Joel Spencer, and Katalin Vesztergombi. Discrepancy of set-systems and matrices. *European Journal of Combinatorics*, 7(2):151–160, 1986.

12 Vivek Madan, Aleksandar Nikolov, Mohit Singh, and Uthaipon Tantipongpipat. Maximizing determinants under matroid constraints. *arXiv preprint*, 2020. arXiv:2004.07886.

13 Vivek Madan, Mohit Singh, Uthaipon Tantipongpipat, and Weijun Xie. Combinatorial algorithms for optimal design. In *Conference on Learning Theory*, pages 2210–2258, 2019.

14 Jiří Matoušek. The determinant bound for discrepancy is almost tight. *Proceedings of the American Mathematical Society*, 141(2):451–460, 2013.

15 Jiří Matoušek, Aleksandar Nikolov, and Kunal Talwar. Factorization norms and hereditary discrepancy. *International Mathematics Research Notices*, 2020(3):751–780, 2020.

16 Kazuo Murota. *Discrete convex analysis*. SIAM, 2003.

17 Aleksandar Nikolov. Randomized rounding for the largest simplex problem. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 861–870, 2015.

18 Aleksandar Nikolov and Mohit Singh. Maximizing determinants under partition constraints. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 192–201, 2016.

19 Aleksandar Nikolov and Kunal Talwar. Approximating hereditary discrepancy via small width ellipsoids. In *Proceedings of the twenty-sixth annual ACM-SIAM symposium on Discrete algorithms*, pages 324–336. SIAM, 2014.

20 Asa Packer. Polynomial-time approximation of largest simplices in v-polytopes. *Discrete Applied Mathematics*, 134:213–237, January 2004. doi:10.1016/S0166-218X(03)00226-9.

21 Dömötör Pálvölgyi. Indecomposable coverings with concave polygons. *Discrete & Computational Geometry*, 44(3):577–588, 2010.

22 Damian Straszak and Nisheeth K Vishnoi. Real stable polynomials and matroids: Optimization and counting. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 370–383, 2017.

Approximating Star Cover Problems

Buddhima Gamlath
École Polytechnique Fédérale de Lausanne, Switzerland
buddhima.gamlath@epfl.ch

Vadim Grinberg
Toyota Technological Institute at Chicago, Chicago, IL, USA
vgm@ttic.edu

Abstract

Given a metric space $(F \cup C, d)$, we consider star covers of C with balanced loads. A star is a pair (i, C_i) where $i \in F$ and $C_i \subseteq C$, and the load of a star is $\sum_{j \in C_i} d(i, j)$. In minimum load k-star cover problem (MLkSC), one tries to cover the set of clients C using k stars that minimize the maximum load of a star, and in minimum size star cover (MSSC) one aims to find the minimum number of stars of load at most T needed to cover C, where T is a given parameter.

We obtain new bicriteria approximations for the two problems using novel rounding algorithms for their standard LP relaxations. For MLkSC, we find a star cover with $(1 + O(\varepsilon))k$ stars and $O(1/\varepsilon^2)\mathrm{OPT}_{\mathrm{MLk}}$ load where $\mathrm{OPT}_{\mathrm{MLk}}$ is the optimum load. For MSSC, we find a star cover with $O(1/\varepsilon^2)\mathrm{OPT}_{\mathrm{MS}}$ stars of load at most $(2 + O(\varepsilon))T$ where $\mathrm{OPT}_{\mathrm{MS}}$ is the optimal number of stars for the problem. Previously, non-trivial bicriteria approximations were known only when $F = C$.

2012 ACM Subject Classification Theory of computation

Keywords and phrases star cover, approximation algorithms, lp rounding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.57

Category APPROX

Related Version A full version of the paper is available at `https://arxiv.org/pdf/1912.01195`.

Acknowledgements We are very grateful to Ola Svensson for influential discussions at multiple stages of this work.

1 Introduction

Facility location (FL) is a family of problems in computer science where the general goal is to assign a set of clients to a set of facilities under various constraints and optimization criteria. FL family encompasses many natural clustering problems like k-median and k-means, most of which are well studied. In this work, we study two relatively less studied FL problems which we call minimum load k-star cover (MLkSC) and minimum size star cover (MSSC). The goal of MLkSC is to assign clients to at most k facilities, minimizing the maximum assignment cost of a facility, while that of MSSC is to find a client-facility assignment with the minimum number of facilities such that the total assignment cost of each facility is upper bounded by a given threshold T.

We begin by formally defining the two problems. Let C be a finite set of clients and F be a finite set of facilities. Let $(F \cup C, d)$ be a finite metric space where $d : (F \cup C) \times (F \cup C) \to \mathbb{R}_0^+$ is a distance metric. By a *star* in (F, C), we mean any tuple (i, C_i), where $i \in F$ and $C_i \subseteq C$. We say two stars (i, C_i) and (h, C_h) are *disjoint* if $i \neq h$ and $C_i \cap C_h = \varnothing$. A *star cover* of (F, C) is a finite collection $S = \{(i_1, C_{i_1}), \ldots, (i_{|S|}, C_{i_{|S|}})\}$ of disjoint stars such that $C = C_{i_1} \cup \cdots \cup C_{i_{|S|}}$. The size of a star cover S is the number of stars $|S|$ in the cover. Given a star cover S, a star $(i, C_i) \in S$, and a client $j \in C_i$, we say that client j is *assigned* to facility i under S and the facility i is *serving* client j under S. For a star (i, C_i), the *load* of facility i


© Buddhima Gamlath and Vadim Grinberg;
licensed under Creative Commons License CC-BY



is the sum of pair-wise distances $\sum_{j \in C_i} d(i,j)$ between itself and its clients. The *load* $L(S)$ of a star cover S is the load of its maximum load star. I.e., $L(S) \coloneqq \max_{(i,C_i) \in S} \sum_{j \in C_i} d(i,j)$. For notational convenience, we denote the collection of all star covers of (F,C) by $\mathcal{S}$. Using the introduced notation, we now define MLkSC and MSSC.

▶ **Definition 1** (Minimum Load k-Star Cover). *Given a finite metric space $(F \cup C, d)$ and number $k \in \mathbb{N}$, the task of minimum load k-star cover problem is to find a star cover of size at most k that minimizes the load; I.e., find $S^* \coloneqq \operatorname{argmin}_{S \in \mathcal{S}: |S| \leq k} L(S)$. We denote the optimal load $L(S^*)$ by $\mathrm{OPT}_{\mathrm{MLk}}$.*

▶ **Definition 2** (Minimum Size Star Cover). *Given a finite metric space $(F \cup C, d)$ and a number $T \in \mathbb{R}_+$, the task of minimum size star cover problem is to find a star cover of load at most T that minimizes the size; I.e., find a star cover $S^\star \coloneqq \operatorname{argmin}_{S \in \mathcal{S}: L(S) \leq T} |S|$. We denote the optimal size $|S^\star|$ by $\mathrm{OPT}_{\mathrm{MS}}$.*

Even et al. [5] showed that both MLkSC and MSSC are NP-hard for general metrics even when $F = C$. Both Even et al. [5] and Arkin et al. [3] studied the problem in $F = C$ setting and gave constant factor bicriteria approximation algorithms for MLkSC. The latter work also gave a constant factor approximation algorithm for MSSC in the same setting.

Arkin et al. [3] use k-median clustering and then split the individual clusters that are too large into several smaller clusters to obtain their approximation guarantees. However, the splitting of clusters rely on that the clients and facilities are indistinguishable, which allows one to conveniently choose a new facility for each new partition created in the splitting process. Meanwhile, the technique of Even et al. [5] is to formulate the problem as an integer program, round its LP relaxation using minimum make-span rounding techniques, and use a clustering approach that also relies on F being equal to C to obtain the final bicriteria approximation guarantees. Both the techniques do not generalize to the case where $F \neq C$ unless it is allowed to open the same facility multiple times.

Recently, Ahmadian et al. [1] showed that MLkSC is NP-hard even if we restrict the metric space to be a line metric. They further gave a PTAS for MLkSC in line metrics and a quasi-PTAS for the same in tree metrics. However, their techniques are specific to line and tree metrics, and it is not known whether they can be extended to general metrics.

The main goal of this work is to extend the approach of Even et al. [5] to $F \neq C$ setting where any given facility can be opened at most once. To do so, we introduce a novel clustering technique and an accompanied new algorithm to modify the LP solution before applying the minimum makespan rounding at the end. This yields the following theorem:

▶ **Theorem 3.** *There exists a polynomial time algorithm that, given an instance $(F \cup C, d)$ of* MLkSC *problem and any $\varepsilon \in (0,1)$, finds a star cover of (F,C) of size at most $(1+O(\varepsilon))k$ and load at most $O(\mathrm{OPT}_{\mathrm{MLk}}/\varepsilon^2)$.*

As a complementary result, we also show that the standard LP relaxation has some inherent limitations. That is, we construct a family of MLkSC instances where the load of any integral $(1+\varepsilon)k$-star cover is at least $\Omega(1/\varepsilon)$ times the optimal value of the standard LP.

With slight modifications to our clustering and rounding techniques, we further obtain the following theorem on MSSC:

▶ **Theorem 4.** *There exists a polynomial time algorithm that, given an instance of* MSSC *problem with load parameter T and any $\varepsilon \in (0,1)$, finds a star cover of load at most $(2+O(\varepsilon))T$ and size at most $O(\mathrm{OPT}_{\mathrm{MS}}/\varepsilon^2)$.*

As with MLkSC, we show that the standard LP-relaxation for MSSC also suffers from inherent limitations; I.e., for any $\varepsilon > 0$, we give an instance of MSSC for which there is a *fractional* star cover of load at most T but any integral star cover of that instance has load at least $(2-\varepsilon)T$ even with all facilities opened.

We end the introduction with a brief section on other related work. In Section 2, we introduce the LP relaxations of the two problems and provide a more elaborate description of our techniques. Later in Section 3 and Section 4 we describe the proofs of Theorem 3 and Theorem 4 in detail. We present the explicit constructions of families of MLkSC and MSSC that show inherent limitations of the respective standard LP relaxations in Appendix B.

Other Related Work

To the best of our knowledge, Even et al. [5] and Arkin et al. [3] were among the first to explicitly address close relatives of MLkSC and MSSC problems. Both of their works considered the problem where one has to cover nodes (or edges) of a graph using a collection of *objects* (I.e.,trees or stars). Even et al. considered the problem of minimizing the maximum cost of an object when the number of objects is fixed, for which they gave a 4-approximation algorithm. Arkin et al. also studied the same problem and additionally considered paths and walks as covering objects. They further discussed the MSSC version of the problems where the goal is to minimize the number of covering objects such that the cost of each object is at most a given threshold. For min-max tree cover with k trees, Khani and Salavatipour [6] later improved the approximation guarantee to a factor of three.

In general, many well-known facility location problems have constant factor approximation guarantees. For example, for uncapacitated facility location, the known best algorithm (Li et al. [8]) gives an approximation ratio of 1.488. For k-median in general metric spaces, the current best is 2.675 due to Byrka et al. [4], and for k-means in general metric spaces, it is $(9+\varepsilon)$ due to Ahmadian et al. [2]. Remarkably, all these results follows from LP based approaches. A common theme of all these problems is that their objectives are to minimize a summation of costs. I.e., we minimize the sum of distances from clients to their respective closest opened facilities, where in uncapacitated facility location problem, we additionally have the sum of opening costs of the opened facilities. This *min-sum* style objective is in contrast with the min-max style objective of minimum star cover problem which makes it immune to algorithmic approaches that are applicable to other common facility location counterparts.

As discussed, minimum star cover problems are closely related to minimum makespan scheduling and the generalized assignment problem. Two most influential literature in this regard include Lenstra et al. [7] and Shmoys et al. [10].

2 Our Results and Techniques

We start with the LP relaxations of the standard integer program formulations for MLkSC and MSSC. To make the presentation easier, we first define a polytope SC-LP(T,k) such that the *integral* points of SC-LP(T,k) are feasible star covers of load at most T and size at most k.

For $i \in F$, let variable $y_i \in \{0,1\}$ denote whether i'th facility is *opened* (I.e., $y_i = 1$ if and only if there is a star (i,C_i) in the target star cover), and for $(i,j) \in F \times C$, let variable $x_{ij} \in \{0,1\}$ denote whether j'th client is *assigned* to facility i (I.e., $x_{ij} = 1$ if and only if $j \in C_i$ where (i,C_i) is a star in the target star cover). Then the following set of constraints define

SC-LP(T,k):

$$\begin{aligned}
&\sum_{j\in C} d(i,j)\cdot x_{ij} \le T\cdot y_i && \forall i\in F, && (1)\\
&\sum_{i\in F} y_i \le k, && && (2)\\
&\sum_{i\in F} x_{ij} = 1 && \forall j\in C, && (3)\\
&x_{ij}\le y_i && \forall i\in F,\forall j\in C, && (4)\\
&y_i\in[0,1] && \forall i\in F, && (5)\\
&x_{ij}\in[0,1] && \forall i\in F,\forall j\in C && (6)\\
&x_{ij}=0 && \forall i\in F,\forall j\in C : d(i,j)>T. && (7)
\end{aligned} \qquad \text{(SC-LP}(T,k)\text{)}$$

Here, Constraint (1) ensures that the load of an opened facility $i \in F$ is at most T, while Constraint (2) limits the maximum number of opened facilities to k. Constraint (3) and Constraint (4) ensure that each client is fully assigned and they are only assigned to opened facilities. Finally Constraint (5) and Constraint (6) ensures that the only integral values of x_{ij}'s and y_i's are 0 or 1, while Constraint (7) essentially removes any (i,j) pair from consideration if the distance between them is larger than T.

Note that we can now define the LP for MLkSC as

$$\text{Minimize } T \text{ such that SC-LP}(T,k) \text{ is feasible,} \qquad \text{(MLkSC-LP)}$$

where one can find the minimum such T using the standard binary search technique. Similarly, the LP for MSSC can be stated as

$$\text{Minimize } k \text{ such that SC-LP}(T,k) \text{ is feasible.} \qquad \text{(MSSC-LP)}$$

Recall that k is part of the MLkSC input and T is a part of the MSSC input.

For an arbitrary (not necessarily feasible) solution (x,y) to SC-LP(T,k), for $i \in F$ let $L(i,x)$ denote the *fractional load* of facility i with respect to the assignment x, I.e., $L(i,x) \coloneqq \sum_{j\in C} d(i,j)x_{ij}$. A solution (x,y) to SC-LP(T,k) is called (α,β)-*approximate*, if for every $i \in F$, $L(i,x) \le \alpha T y_i$, and $\sum_{i\in F} y_i \le \beta k$. The proofs of Theorem 3 and Theorem 4 immediately follow from the two theorems on rounding feasible solutions of SC-LP presented below:

▶ **Theorem 5.** *There exists a polynomial time rounding algorithm that, given a feasible solution (x^*,y^*) to* SC-LP(T,k) *and any $\varepsilon \in (0,1)$, outputs an integral $(O(1/\varepsilon^2), 1+O(\varepsilon))$-approximate solution to* SC-LP(T,k).

Proof of Theorem 3. Let $\varepsilon \in (0,1)$ be given. Using standard binary search approach, we can guess the value T^*, such that $\mathrm{OPT}_{\mathrm{MLk}} \le T^* \le 2\mathrm{OPT}_{\mathrm{MLk}}$, by solving MLkSC-LP multiple times for different values of T^* and either finding a feasible fractional solution of load at most T^*, or determining that no such solution exists. Let (x^*,y^*) be the corresponding fractional solution to MLkSC-LP. Observe that (x^*,y^*) is a feasible solution to SC-LP(T^*,k). By Theorem 5, we can round (x^*,y^*) to an integral solution $(\dot{x},\dot{y})$, which opens at most $(1+O(\varepsilon))k$ facilities and achieves maximum load at most $O(1/\varepsilon^2)T^*$, and it will take polynomial time. Therefore, $(\dot{x},\dot{y})$ will be an integral solution to MLkSC-LP with opening at most $(1+O(\varepsilon))k$ and maximum load at most $O(1/\varepsilon^2)\mathrm{OPT}_{\mathrm{MLk}}$. ◀

▶ **Theorem 6.** *There exists a polynomial time rounding algorithm that, given a feasible solution* (x^*, y^*) *to* SC-LP(T, k) *and any* $\varepsilon \in (0, 1)$*, outputs an integral* $(2 + O(\varepsilon), O(1/\varepsilon^2))$*-approximate solution to* SC-LP(T, k).

The proof of Theorem 4 using Theorem 6 is just the same as the proof of Theorem 3 using Theorem 5, omitting the binary search part (as we optimize over k instead of T).

Note that MLkSC-LP closely resembles the LP used in minimum make-span rounding by Lenstra et al. [7]. In fact, for the case where we do not have a restriction on number of opened facilities, we can assume $y_i = 1$ for all $i \in F$, and the LP reduces to the minimum make-span problem, yielding a 2-approximation algorithm. The main difficulty here is to figure out which facilities to open. Once we have an integral opening of facilities, we can still use minimum make-span rounding at a loss of only a factor 2 in the guarantee for minimum load. Thus, our algorithm for MLkSC essentially transforms the initial solution for MLkSC-LP via a series of steps to a solution with integral openings, I.e., $y_i \in \{0, 1\}$ for all $i \in F$, and fractional assignments, without violating Constraint 1 by too much.

When we fully open (I.e., set $y_i = 1$) some facilities in the solution, inevitably, we have to close down (set $y_i = 0$) some other partially opened facilities, which requires redistributing their assigned clients' demand to the opened ones. This process is called *rerouting* and is a well-known technique in rounding facility-location-like problems. However, instead of bounding the total load of all facilities, our problem requires bounding each $L(i, x)$ for $i \in F$ separately, and consequently, many facility-location rounding algorithms which use rerouting fail to produce a good solution.

Let x° be the solution we obtain from x after rerouting facility i to facility h. Using triangle inequality $d(h, j) \le d(h, i) + d(i, j)$ for $j \in C$, we bound $L(h, x^\circ)$, the new load of h:

$$L(h, x^\circ) \le L(h, x) + L(i, x) + d(h, i) \sum_{j \in C} x_{ij}.$$

If both $L(h, x)$ and $L(i, x)$ were initially $O(T)$, the new load of h will also be $O(T)$ if and only if the value $d(h, i) \sum_{j \in C} x_{ij} \le d(h, i)|N(i)|$ is also at most $O(T)$ (here $N(i)$ is the set of all clients partially served by i). However, if $d(h, i)|N(i)|$ is large for all other facilities h, a good alternative to rerouting is to open i integrally and assign every client in $N(i)$ to i. We call such facilities *heavy* facilities. There is still an issue if the integral load $\sum_{j \in N(i)} d(i, j)$ is too large compared to T, but we show that we can prevent having too large integral loads in heavy facilities by preceding the rerouting step with additional filtering and preprocessing steps. The filtering step blows-up the load constraint by a $(1 + \varepsilon)$ factor while ensuring that no client is fractionally assigned to far away facilities. The preprocessing step uses techniques similar to those of minimum make-span rounding by Lenstra et al. [7] to ensure that any non-zero fractional assignment x_{ij} to a facility i is at least a constant factor times its opening y_i, while slightly relaxing other constraints.

Once we identify the heavy facilities, we cluster the remaining, non-heavy facilities, and choose which ones should be opened based on the clustering. Then we redistribute the assignments of the remaining facilities to those that were opened. Using the properties of the preprocessed solution and the clustering, and using the fact that none of the non-opened facilities are heavy, we show that the resulting fractional assignment satisfies the constraints up to an $O(1/\varepsilon^2)$ factor violation of load constraints. Hence, the algorithmic result of Theorem 5 follows from the minimum make-span rounding of Lenstra et al. [7], which gives us an integral assignment with maximum load increased at most by another factor of 2

The algorithm for MSSC problem, on a high level, resembles that for MLkSC: We first alter the solution of MSSC-LP to have integral y_i's and fractional x_{ij}'s, allowing the total opening $\sum_{i \in F} y_i$ to be at most $O(1/\varepsilon^2)$ factor larger than the value of MSSC-LP, and then

use minimum make-span rounding of Lenstra et al. [7] to obtain the final solution. However, since make-span rounding guarantees only a factor 2 violation in the load constraint, we need to make sure that our modified solution with integral openings and fractional assignments introduces only small error in load constraints. Namely, to ensure that the final solution satisfies $(2+O(\varepsilon))T$ maximum load, before applying the minimum make-span rounding, all the loads must be at most $(1+O(\varepsilon))T$. We ensure this by re-arranging the steps of the algorithm for MLkSC and carefully choosing parameters.

3 $(O(1/\varepsilon^2), 1+O(\varepsilon))$-approximation to SC$(T,k)$

In this section, we show how to convert a (feasible) fractional solution (x,y) of SC-LP in to a $(O(1/\varepsilon^2), 1+O(\varepsilon))$-approximate solution with integral y values. This together with minimum make-span rounding scheme by Lenstra et al. [7] proves Theorem 5.

3.1 Preprocessing and filtering

Suppose that for each $(i,j) \in F \times C$ we either have $x_{ij} = 0$ or $x_{ij} \geq \gamma y_i$ for constant $\gamma \in (0,1)$. Then, if $L(i,x) = \sum_{j \in C} d(i,j)x_{ij} \leq \nu T y_i$ for some constant $\nu \geq 1$, we have $\sum_{j \in N(i)} d(i,j) \leq \frac{\nu}{\gamma} T$. Therefore, if we open i integrally and assign all $N(i)$ to i, the resulting load of i will be $O(T)$. Even though we cannot guarantee the property above for every solution (x,y) to SC-LP(T,k), we can modify (x,y) so that all non-zero assignments x_{ij} satisfy $x_{ij} \geq \gamma y_i$ for some constant $\gamma \in (0,1)$ at the expense of slightly relaxing other constraints of SC-LP. This is exactly the statement of the preprocessing theorem.

▶ **Theorem 7** (Preprocessing). *Let (x,y) be such that, for all $i \in F$, $L(i,x) \leq \mu T y_i$ for some constant $\mu \geq 1$ and all other constraints of* SC-LP(T,k) *on variables x are satisfied. There exists a polynomial time algorithm that, given such solution (x,y) and a constant $\gamma \in (0,1)$, finds a solution (x',y') such that*

1. *$y' = y$, and if $x_{ij} = 0$, then $x'_{ij} = 0$;*
2. *for every $(i,j) \in F \times C$, $y'_i \geq x'_{ij}$, and if $x'_{ij} > 0$, then $x'_{ij} \geq \gamma y'_i$;*
3. *for every $j \in C$, $1 \geq \sum_{i \in F} x'_{ij} \geq 1-\gamma$;*
4. *for every $i \in F$, $L(i,x') \leq (\mu + 2 - \gamma) T y'_i$.*

That is to say, we can guarantee the property $\{x_{ij} > 0 \iff x_{ij} \geq \gamma y_i\}$ by loosing at most γ portion of each client's demand and slightly increasing each facility's load. Loosing a factor of γ demand is affordable for our purposes, as one can meet the demand constraint by scaling each x_{ij} by a factor of at most $1/(1-\gamma)$. Since γ is a constant, this would blow up the load constraint only by an additional constant factor. The proof of Theorem 7 is rather technical and is given in Appendix A.

We now present our rounding algorithm step by step. Let (x,y) be a feasible fractional solution to SC-LP(T,k) and let $\varepsilon \in (0,1)$. By $(\dot{x},\dot{y})$ we denote the final rounded solution with integral $\dot{y}$ and fractional $\dot{x}$.

▶ **Definition 8.** *For $j \in C$, let $D(j) := \sum_{i \in F} d(i,j)x_{ij}$, the average facility distance to j.*

Let $\rho := \frac{1+\varepsilon}{\varepsilon}$. By applying the well-known filtering technique of Lin and Vitter [9] to (x,y), we construct a new solution $(\hat{x},\hat{y})$, such that $\sum_{i \in F} \hat{y}_i \leq (1+\varepsilon)k$, $L(i,\hat{x}) \leq (1+\varepsilon)T\hat{y}_i$ for all $i \in F$, and for every i,j, $\hat{x}_{ij} \leq \hat{y}_i$ and if $\hat{x}_{ij} > 0$, then $d(i,j) \leq \rho D(j)$. Applying Theorem 7 to $(\hat{x},\hat{y})$, we obtain solution (x',y') such that

1. $\sum_{i \in F} y'_i \leq (1+\varepsilon)k$,
2. for all (i,j), $y'_i \geq x'_{ij}$, and if $x'_{ij} > 0$, then $x'_{ij} \geq \gamma y'_i$ and $d(i,j) \leq \rho D(j)$,

3. for every $j \in C$, $1 \geq \sum_{i \in F} x'_{ij} \geq 1 - \gamma$, and
4. for every $i \in F$, $L(i, x') \leq (\mu + 2 - \gamma) T y'_i = \nu T y'_i$.

Here $\nu := (\mu + 2 - \gamma)$ is a new load bound. We choose $\mu := 1 + \varepsilon$ and $\gamma := \varepsilon/(1+\varepsilon)$, but will keep the parameters unsubstituted, for convenience. It is easy to see from the bounds above that for every $j \in C$, $\sum_{i \in F : x'_{ij} > 0} y'_i \geq 1/(1+\varepsilon)$.

3.2 Opening heavy facilities

We now give an algorithm to choose heavy facilities based on (x', y').

▶ **Definition 9.** *For subsets $F' \subseteq F$, $C' \subseteq C$, for $i \in F'$ denote $N'(i) := \{j \in C' : x'_{ij} > 0\}$, and for $j \in C'$ denote $N'(j) := \{i \in F' : x'_{ij} > 0\}$.*

The algorithm internally maintains two subsets $F' \subseteq F$ and $C' \subseteq C$. Notice that N' changes as the algorithm modifies F' and C'.

▶ **Definition 10.** *A facility $i \in F'$ is* λ-heavy *for $\lambda > 0$, if $\sum_{j \in N'(i)} D(j) > \lambda T$.*

Algorithm 1 opens all λ-heavy facilities for the given value of λ. It starts with $F' = F$ and $C' = C$ and scans F' for λ-heavy facilities. It fully opens every λ-heavy facility $i \in F'$ and assigns all $N'(i)$ integrally to i. Then, it discards i from F' and $N'(i)$ from C', and continues until all facilities are processed.

Algorithm 1 Opening Heavy Facilities.

Input: A solution (x', y'), $\lambda > 0$.
Output: Partial solution $(\dot{x}, \dot{y})$, sets F', C', such that $\sum_{j \in N'(i)} D(j) \leq \lambda T$, $\forall i \in F'$.

```
Initialize F' ← F, C' ← C
for i ∈ F' do
    if Σ_{j∈N'(i)} D(j) > λT then
        Initialize C(i) ← N'(i)
        F' ← F' ∖ {i}, ẏ_i ← 1
        for j ∈ C(i) do
            C' ← C' ∖ {j}, ẋ_ij ← 1
            for h ∈ F ∖ {i} do ẋ_hj ← 0
return (ẋ, ẏ), F', C'.
```

Since for each $h \in F'$ we may discard some clients from $N'(h)$ after every step, facilities that were λ-heavy might become non-λ-heavy under updated F' and C'. Lemma 11 shows that this procedure does not open too many facilities and that the load of opened facilities does not exceed T by too much.

▶ **Lemma 11.** *Let F', C' be the sets returned by Algorithm 1. Then $|F \setminus F'| \leq k/\lambda$, and for each facility $i \in F \setminus F'$, $L(i, \dot{x}) \leq \frac{\nu}{\gamma} T$.*

Proof. The set $F \setminus F'$ is exactly the set of facilities integrally opened during Algorithm 1. For $i \in F \setminus F'$, set $C(i)$ in Algorithm 1 is exactly the set of clients, integrally assigned to i by the algorithm. Observe that for every $i, h \in F \setminus F'$, $i \neq h$, the sets $C(i)$ and $C(h)$ are *disjoint*. Hence, by feasibility of (x, y),

$$|F \setminus F'| \cdot \lambda T < \sum_{i \in F \setminus F'} \sum_{j \in C(i)} D(j) \leq \sum_{j \in C} D(j) = \sum_{j \in C} \sum_{i \in F} d(i,j) x_{ij} \leq \sum_{i \in F} T y_i \leq Tk$$

and $|F \setminus F'| < \frac{Tk}{\lambda T} = \frac{k}{\lambda}$. Next, by the properties of solution (x', y'), $\nu T y'_i \geq \sum_{j \in C(i)} d(i,j) x'_{ij} \geq \gamma y'_i \sum_{j \in C(i)} d(i,j)$, implying $L(i, \dot{x}) = \sum_{j \in C(i)} d(i,j) \leq \frac{\nu}{\gamma} T$. ◀

We apply Algorithm 1 with $\lambda := 1/\varepsilon$, and by Lemma 11 this opens at most εk additional facilities. The load of each opened facility is at most $\frac{\nu}{\gamma}T = \frac{(1+\varepsilon)(\mu+2-\gamma)}{\varepsilon}T = O(T/\varepsilon)$. For the returned sets F' and C', $\sum_{j\in N'(i)} D(j) \le T/\varepsilon$ for all $i \in F'$. Moreover, since $j \in C'$ if and only if j was not served by any λ-heavy facility (which got opened), for all $j \in C'$ we have $\sum_{i\in N'(j)} y'_i = \sum_{i\in F: x'_{ij}>0} y'_i \ge 1/(1+\varepsilon)$. Facilities in $F \setminus F'$ are all integral, and it remains to find the integral opening among facilities in F'.

As discussed earlier, if we reroute $i \in F'$ to $h \in F'$, to guarantee a good approximation we have to bound the term $d(h,i)|N'(i)|$. Observe that $\sum_{j\in N'(i)} D(j)$ is an upper bound for $|N'(i)| \min_{j\in N'(i)} D(j)$. Therefore, to get a good bound, we need to choose h for i so that $d(h,i)$ is at most some constant times $\min_{j\in N'(i)} D(j)$. This requires some sophisticated clustering technique and a wise choice of facility h for every such i.

3.3 Clustering

To create an integral opening over F', we partition F' into disjoint clusters, open some facilities in every cluster and reroute the closed ones into opened ones within the same cluster. Our goal is to cluster F' so that, if i and h belong to the same cluster and we reroute h to i, $d(h,i) \le O(\min_{j\in N'(i)} D(j))$. Classic clustering approaches for facility-location-like problems do not work, and to achieve this bound we are required to design a novel approach.

Let $\mathcal{C} \subseteq C'$ be the set of cluster centers. For every $j \in \mathcal{C}$, let $F'(j) \subseteq F'$ be the set of facilities belonging to the cluster centered at j, for $i \in F'$ let $\mathcal{C}(i)$ be the center of the cluster i belongs to (I.e., $i \in F'(j) \iff \mathcal{C}(i) = j$). Figure 1 visualizes the clustering procedure and Algorithm 2 gives its pseudocode. Below we explain each step of the algorithm in detail.

Algorithm 2 Clustering.

Input: Solution (x', y'), sets F' and C'.
Output: Centers $\mathcal{C} \subseteq C'$ and disjoint clusters $F'(s)$ for every $s \in \mathcal{C}$, $\cup_{s\in\mathcal{C}} F'(s) = F'$.

Initialize $\mathcal{C} \leftarrow \varnothing$, sort $j \in C'$ by the values of $D(j)$ in ascending order
for $j \in C'$ **do**
  if $\forall s \in \mathcal{C} : d(s,j) > 2\rho D(j)$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} \cup \{j\}$
For all $s \in \mathcal{C}$, initialize $F'(s) \leftarrow \varnothing$
for $i \in F'$ **do**
  if $\exists s \in \mathcal{C} : i \in N'(s)$ **then**
    $\mathcal{C}(i) \leftarrow s$, $F'(s) \leftarrow F'(s) \cup \{i\}$
  else
    Let $j := \operatorname{argmin}_{r\in N'(i)} D(r)$
    Take $s \in \mathcal{C}$, such that $D(s) \le D(j)$, $d(s,j) \le 2\rho D(j)$
    $\mathcal{C}(i) \leftarrow s$, $F'(s) \leftarrow F'(s) \cup \{i\}$
return $\mathcal{C}$, $F'(s)$ for $s \in \mathcal{C}$.

The clustering procedure works as follows. First, we form cluster centers $\mathcal{C}$ by scanning $j \in C'$ in ascending order of $D(j)$ and adding j to $\mathcal{C}$ only if there are no other centers in $\mathcal{C}$ within the distance $2\rho D(j)$ from j. Having determined $\mathcal{C}$, we add facilities from F' to different clusters. Most classical clustering approaches would put i into $F'(s)$, if s is closest to i among $\mathcal{C}$. Our approach is different: if $i \in F'$ is serving some $s \in \mathcal{C}$, we add $i \in F'(s)$ regardless the distance $d(i,s)$. Otherwise, we consider $j \in N'(i)$ with *minimum* $D(j)$ (j is not a cluster center), take $s \in \mathcal{C}$ that prevented j from becoming a center, and add i to $F'(s)$. One can easily check that, after Algorithm 2 finishes, for any $s, v \in \mathcal{C}$,

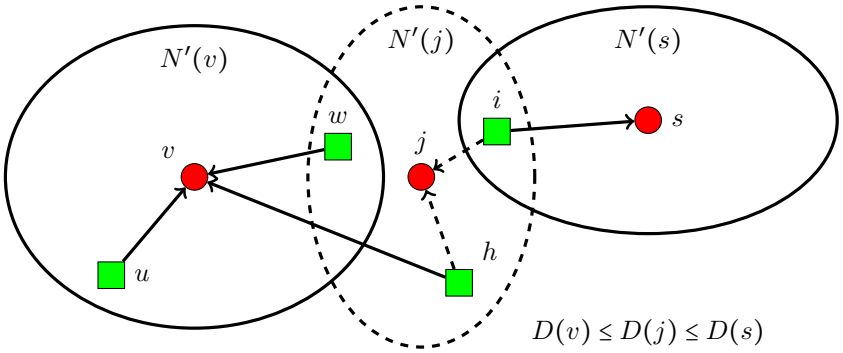


Figure 1 Here, $v, j, s \in C'$, $v, s \in \mathcal{C}$, $j \notin \mathcal{C}$, $u, w, h, i \in F'$. The bold arrow $\rightarrow$ shows that a facility belongs to the cluster centered at that client, the dashed arrow $\dashrightarrow$ shows that a particular client has minimum average distance among all clients served by a facility.

$s \neq v$, $d(s,v) > 2\rho \max\big(D(s), D(v)\big)$, and as a result $N'(s)$ and $N'(v)$, as well as $F'(s)$ and $F'(v)$ are disjoint. Also, for every $j \notin \mathcal{C}$ there exists $s \in \mathcal{C}$ such that $D(s) \le D(j)$ and $d(s,j) \le 2\rho D(j)$, simply by construction of the algorithm. Algorithm 2 allows us to obtain an upper bound on the distance between a facility an its cluster center, represented in terms of minimum average distance of the client served by this facility.

▶ **Lemma 12.** *Let $i \in F'$, let $j = \operatorname{argmin}_{r \in N'(i)} D(r)$. Then $d(i, \mathcal{C}(i)) \le 3\rho D(j)$.*

Proof. Let $\mathcal{C}(i) = s$. There are two cases to distinguish.

- $i \notin N'(s)$ (this case is shown by clients j, v and facility h in Figure 1). By construction of Algorithm 2, client s is exactly the one that prevented j from becoming a cluster center, therefore $D(s) \le D(j)$ and $d(j,s) \le 2\rho D(j)$. Thus, by triangle inequality $d(i,s) \le d(i,j) + d(j,s) \le \rho D(j) + 2\rho D(j) = 3\rho D(j)$.
- $i \in N'(s)$. Then $D(j) = \min_{r \in N'(i)} D(r) \le D(s)$. If $s = j$ or $D(s) = D(j)$, then $d(i,s) \le \rho D(j)$ automatically. Suppose that $s \neq j$, and $D(j) < D(s)$, then $j \notin \mathcal{C}$, as $s \in \mathcal{C}$ and $i \in N'(j) \cap N'(s)$ (this case is shown by clients j, s and facility i in Figure 1). Hence, there exists some $s' \in \mathcal{C}$ that prevented j from becoming a cluster center, so $D(s') \le D(j)$ and $d(s', j) \le 2\rho D(j)$. It is easy to see that $D(j)$ must be strictly greater than zero, and since both s and s' are cluster centers, $d(s', s) > 2\rho D(s)$. So, by triangle inequality,

$$2\rho D(s) < d(s', s) \le d(s', j) + d(i, j) + d(i, s) \le 2\rho D(j) + \rho D(j) + \rho D(s),$$

  implying $2\rho D(s) \le 3\rho D(j) + \rho D(s)$ and $D(s) \le 3D(j)$. Since $i \in N'(s)$, it immediately follows that $d(i,s) \le \rho D(s) \le 3\rho D(j)$. ◀

By applying the triangle inequality once more, we get the desired upper bound on the distances between any two facilities within the same cluster.

▶ **Corollary 13.** *Let $i, h \in F'$, such that $\mathcal{C}(i) = \mathcal{C}(h)$. Let $j = \operatorname{argmin}_{r \in N'(i)} D(r)$ and $v = \operatorname{argmin}_{l \in N'(h)} D(l)$. Then $d(i,h) \le 6\rho \max\big(D(j), D(v)\big)$.*

Another useful observation is that $\sum_{i \in F'(s)} y'_i \ge 1/(1+\varepsilon)$ for every cluster center $s \in \mathcal{C}$. It follows from $N'(s) \subseteq F'(s)$ and $\sum_{i \in N'(s)} y'_i \ge 1/(1+\varepsilon)$.

3.4 Rerouting

The last part of our rounding algorithm is opening some facilities in every cluster and rerouting the closed ones. For $s \in \mathcal{C}$ we open $\lfloor (1+\varepsilon) \sum_{u \in F'(s)} y'_u \rfloor$ facilities in cluster $F'(s)$, prioritizing facilities i with minimum values of $\min_{r \in N'(i)} D(r)$. Since $\sum_{u \in F'(s)} y'_u \ge 1/(1+\varepsilon)$,

we will open at least one facility in every cluster $F'(s)$ for $s \in \mathcal{C}$. Then, the demand of each closed facility in $F'(s)$ is redistributed at an equal fraction between all the opened ones in $F'(s)$, I.e. we reroute it to all opened facilities in $F'(s)$. This gives us an integral opening $\dot{y}$ over facilities in F' and a fractional assignment $\dot{x}$ over clients in C'.

Lemma 14 shows that by opening $|K_s| = \lfloor (1+\varepsilon) \sum_{u \in F'(s)} y'_u \rfloor$ facilities in cluster $F'(s)$ and rerouting all closed facilities in $F'(s)$, we open at most $(1+3\varepsilon)k$ facilities in total, and the load of every opened facility in F' exceeds T at most by a constant factor.

Algorithm 3 Rerouting.

Input: Solution (x', y'), cluster centers $\mathcal{C}$ and clusters $F'(s)$ for $s \in \mathcal{C}$.
Output: Solution $(\dot{x}, \dot{y})$, sets of opened facilities K_s for $s \in \mathcal{C}$.

for $s \in \mathcal{C}$ **do**
  Initialize $K_s \leftarrow \varnothing$
  Sort $i \in F'(s)$ in ascending order of $\min_{r \in N'(i)} D(r)$
  for $i \in F'(s)$ **do**
    if $y'_i = 0$ **then**
      $\dot{y}_i \leftarrow 0$
    else if $|K_s| + 1 \le \lfloor (1+\varepsilon) \sum_{u \in F'(s)} y'_u \rfloor$ **then**
      $K_s \leftarrow K_s \cup \{i\}$, $\dot{y}_i \leftarrow 1$
      for $j \in N'(i)$ **do** $\dot{x}_{ij} \leftarrow x'_{ij}$
    else
      $\dot{y}_i \leftarrow 0$
      for $r \in N'(i)$ **do**
        $\dot{x}_{ir} \leftarrow 0$
        for $h \in K_s$ **do**
          $\dot{x}_{hr} \leftarrow \dot{x}_{hr} + x'_{ir}/|K_s|$
return $(\dot{x}, \dot{y})$, K_s for $s \in \mathcal{C}$.

▶ **Lemma 14.** *After Algorithm 3, for every facility $h \in F'$, $\dot{y}_h = 1$: $L(h, \dot{x}) \le 3(\nu + 4\rho\lambda)T$. Moreover, $\sum_{h \in F'} \dot{y}_h \le (1+3\varepsilon)k$.*

Proof. Since for every $s \in \mathcal{C}$ we have $\sum_{u \in F'(s)} y'_u \ge 1/(1+\varepsilon)$, $\lfloor (1+\varepsilon) \sum_{u \in F'(s)} y'_u \rfloor \ge 1$ and $|K_s| \ge 1$. After filtering and preprocessing steps, $\sum_{u \in F'} y'_u \le (1+\varepsilon)k$, so

$$\sum_{h \in F'} \dot{y}_h = \sum_{s \in \mathcal{C}} \sum_{h \in K_s} \dot{y}_h \le (1+\varepsilon) \sum_{s \in \mathcal{C}} \sum_{u \in F'(s)} y'_u = (1+\varepsilon) \sum_{u \in F'} y'_u \le (1+\varepsilon)^2 k \le (1+3\varepsilon)k.$$

Next, let $h \in F'$, $\dot{y}_h = 1$, and let $\mathcal{C}(h) = s$. Take $i \in F'(s)$ that was closed by Algorithm 3. The demand of every $r \in N'(i)$ served by i gets split between all opened facilities from K_s at an equal fraction. So, after we reroute i into h, the *additional* load of h is

$$\sum_{r \in N'(i)} d(h,r) \frac{x'_{ir}}{|K_s|} \le \frac{1}{|K_s|} \sum_{r \in N'(i)} d(i,r) x'_{ir} + \frac{d(h,i)}{|K_s|} \sum_{r \in N'(i)} x'_{ir}.$$

Recall that $\sum_{r \in N'(i)} d(i,r) x'_{ir} = L(i, x') \le \nu T y'_i$. Let $v = \operatorname{argmin}_{l \in N'(h)} D(l)$ and $j = \operatorname{argmin}_{r \in N'(i)} D(r)$. Since h was opened, and i was closed, $D(v) \le D(j)$, and by Corollary 13 $d(h,i) \le 6\rho D(j)$. Hence, the *additional* load of h is at most

$$\frac{1}{|K_s|}\sum_{r\in N'(i)} d(i,r)x'_{ir} + \frac{d(h,i)}{|K_s|}\sum_{r\in N'(i)} x'_{ir} \le \frac{\nu T y'_i}{|K_s|} + \frac{6\rho D(j)}{|K_s|}\sum_{r\in N'(i)} x'_{ir} \le$$
$$\le \frac{\nu T y'_i}{|K_s|} + \frac{6\rho D(j)}{|K_s|}\sum_{r\in N'(i)} y'_i = \frac{y'_i}{|K_s|}\left(\nu T + 6\rho\cdot|N'(i)|D(j)\right) \le$$
$$\le \frac{y'_i}{|K_s|}\left(\nu T + 6\rho\cdot\lambda T\right) = \frac{y'_i}{|K_s|}(\nu + 6\rho\lambda)T.$$

We used the bound $|N'(i)|\min_{r\in N'(i)} D(r) \le \sum_{r\in N'(i)} D(r) \le \lambda T$ for non-λ-heavy facilities. Hence, the *total additional* load of h, gained after rerouting all closed facilities $i \in F'(s)\setminus K_s$ in its cluster, is at most

$$\sum_{i\in F'(s)\setminus K_s}\sum_{r\in N'(i)} d(h,r)\frac{x'_{ir}}{|K_s|} \le \sum_{i\in F'(s)\setminus K_s}\frac{y'_i}{|K_s|}(\nu+6\rho\lambda)T =$$
$$= (\nu+6\rho\lambda)T\cdot\frac{\sum_{i\in F'(s)\setminus K_s} y'_i}{\lfloor(1+\varepsilon)\sum_{u\in F'(s)} y'_u\rfloor} \le (\nu+6\rho\lambda)T\cdot\frac{(1+\varepsilon)\sum_{i\in F'(s)} y'_i}{\lfloor(1+\varepsilon)\sum_{u\in F'(s)} y'_u\rfloor} \le$$
$$\le (2\nu+12\rho\lambda)T.$$

The load of h before rerouting was $L(h,x') \le \nu T y'_h \le \nu T$, so after Algorithm 3 the total load of facility h is $L(h,\dot{x}) \le 3(\nu+4\rho\lambda)T$. This holds for every $h \in K_s$ and every center $s \in \mathcal{C}$. ◀

Now we are ready to complete the analysis of the rounding algorithm.

Proof of Theorem 5. We claim that, having completed all the intermediate steps from filtering and up to Algorithm 3 included, with parameter values $\rho = \frac{1+\varepsilon}{\varepsilon}$, $\gamma = \varepsilon/(1+\varepsilon)$ and $\lambda = 1/\varepsilon$, for the resulting solution $(\dot{x},\dot{y})$ it holds:

1. $\dot{y}$ is integral, and $\sum_{i\in F}\dot{y}_i \le (1+4\varepsilon)k$;
2. for every $j \in C$, $1 \ge \sum_{i\in F}\dot{x}_{ij} \ge 1/(1+\varepsilon)$, and if $j \in C\setminus C'$, $\sum_{i\in F}\dot{x}_{ij} = 1$;
3. for every $i \in F$, $L(i,\dot{x}) \le 12\left(1+\frac{1+\varepsilon}{\varepsilon^2}\right)T\dot{y}_i$.

By Lemma 11, Algorithm 1 could open additional εk facilities, so $\sum_{i\in F\setminus F'}\dot{y}_i \le \varepsilon k$. By Lemma 14, $\sum_{h\in F'}\dot{y}_h \le (1+3\varepsilon)k$. This gives us total opening $\sum_{i\in F}\dot{y}_i \le (1+4\varepsilon)k$.

Next, take $j \in C$. If j was serving some λ-heavy facility i, then $j \in C\setminus C'$, and Algorithm 1 sets $\dot{x}_{ij} = 1$ and $\dot{x}_{hj} = 0$ for all other facilities $h \ne i$. If j did not serve any λ-heavy facility, then $j \in C'$, and we get $1 \ge \sum_{i\in F}\dot{x}_{ij} = \sum_{i\in F} x'_{ij} \ge 1/(1+\varepsilon)$ after rerouting.

Finally, if $i \in F$ was λ-heavy, by Lemma 11 $L(i,\dot{x}) \le \frac{\nu}{\gamma}T \le \frac{4}{\varepsilon}T\dot{y}_i$. Let i be non-λ-heavy, I.e. $i \in F'$. If $\dot{y}_i = 0$, I.e. i is closed, then Algorithm 3 assures that $L(i,\dot{x}) = 0$. If $\dot{y}_i = 1$, then by Lemma 14 we have $L(i,\dot{x}) \le 3(\nu+4\rho\lambda)T \le 3\left(4+4\frac{1+\varepsilon}{\varepsilon^2}\right)T\dot{y}_i = 12\left(1+\frac{1+\varepsilon}{\varepsilon^2}\right)T\dot{y}_i$.

For every $(i,j) \in F'\times C'$, we multiply the assignment variables $\dot{x}_{ij}$ by $1/\left(\sum_{i\in F}\dot{x}_{ij}\right)$. Since $\sum_{i\in F}\dot{x}_{ij} \ge 1/(1+\varepsilon)$, the load of every opened facility in F' gets increased at most by a factor of $1+\varepsilon \le 2$. After this change, $\sum_{i\in F}\dot{x}_{ij} = 1$ and $L(i,\dot{x}) \le 24\left(1+\frac{1+\varepsilon}{\varepsilon^2}\right)T$ for all $j \in C$, $i \in F$.

The solution $(\dot{x},\dot{y})$ has integral opening $\dot{y}$, and every client $j \in C$ is served fully (I.e. $\sum_{i\in F}\dot{x}_{ij} = 1$). By applying minimum makespan rounding algorithm [7], we get an integral assignment with respect to facilities opened in $\dot{y}$, sacrificing another factor of 2 in approximation. We obtain a $\left(48\left(1+\frac{1+\varepsilon}{\varepsilon^2}\right), 1+4\varepsilon\right)$-approximate solution to SC(T,k) problem, and the whole algorithm clearly runs in polynomial-time. ◀

4 $(2+O(\varepsilon), O(1/\varepsilon^2))$-approximation to $\mathrm{SC}(T,k)$

Similar to the $(O(1/\varepsilon^2), 1+O(\varepsilon))$-approximation to $\mathrm{SC}(T,k)$, our goal is, given some solution (x,y) to SC-LP(T,k), find an integral opening $\dot{y}$ and fractional assignment $\dot{x}$, and then apply minimum makespan rounding [7], which will prove Theorem 6. However, this time we need to assure that $L(i,\dot{x}) \le (1+O(\varepsilon))T$ for every $i \in F$. To achieve this, we use the same steps, applied in different order and with different values of parameters.

4.1 Preprocessing and opening heavy facilities

Let (x,y) be a feasible fractional solution to SC-LP(T,k), and let $\varepsilon \in (0,1)$. Straightahead, we apply preprocessing algorithm from Theorem 7 to (x,y) with parameters $\mu = 1$ and $\gamma = \frac{1}{1+\varepsilon}$. This will give us a solution (x',y') such that

1. $y' = y$, and $\sum_{i\in F} y'_i \le k$,
2. for all $(i,j) \in F \times C$, $y'_i \ge x'_{ij}$ and if $x'_{ij} > 0$ then $x'_{ij} \ge \gamma y'_i = y'_i/(1+\varepsilon)$,
3. for every $j \in C$, $1 \ge \sum_{i\in F} x'_{ij} \ge 1-\gamma = \varepsilon/(1+\varepsilon)$, and
4. for every $i \in F$, $L(i,x') \le (\mu+2-\gamma)Ty'_i = (2+\frac{\varepsilon}{1+\varepsilon})Ty'_i = \nu T y'_i$.

In this algorithm, we overuse the notation and define $D(j)$ with respect to assignment x'.

▶ **Definition 15.** *For $j \in C$, let $D(j) := \sum_{i\in F} d(i,j)x'_{ij}$, the average facility distance to j.*

The definitions of $N'(i)$, $N'(j)$ for $i \in F'$, $j \in C'$, given $F' \subseteq F$ and $C' \subseteq C$, are the same.

▶ **Definition 16.** *For $F' \subseteq F$, $C' \subseteq C$, let $N'(i) := \{j \in C' : x'_{ij} > 0\}$, $N'(j) := \{i \in F' : x'_{ij} > 0\}$.*

▶ **Definition 17.** *A facility $i \in F'$ is λ-heavy for $\lambda > 0$, if $\sum_{j\in N'(i)} D(j) > \lambda T$.*

We apply Algorithm 1 to solution (x',y') with $\lambda := \varepsilon^2/15$. Observe that $\sum_{j\in C} D(j) = \sum_{i\in F}\sum_{j\in C} d(i,j)x'_{ij} \le \sum_{i\in F} \nu T y'_i \le \nu T k$. Hence, using a similar analysis as in Lemma 11, we open at most $\frac{\nu}{\lambda}k = O(k/\varepsilon^2)$ additional facilities, and the load of every opened facility is at most $\frac{\nu}{\gamma}T = (1+\varepsilon)\left(2+\frac{\varepsilon}{1+\varepsilon}\right)T = (2+3\varepsilon)T$.

For the returned sets F' and C', $\sum_{j\in N'(i)} D(j) \le \lambda T = \varepsilon^2 T/15$, for every $i \in F'$. As before, it remains to find the integral opening among facilities in F'. However, there may be clients $j \in C'$, for which preprocessing step might have dropped a very huge portion of their demand, as the best bound we have is $\sum_{i\in F'} x'_{ij} \ge \varepsilon/(1+\varepsilon)$. Just for the same reason, the opening $\sum_{i\in N'(j)} y'_i$ may be too small for some clients $j \in C'$, so we cannot apply the clustering and rerouting steps to solution (x',y'), as we did in Section 3, without loosing a lot in both approximation factors, we even do not have any distance upper bounds. We are going to handle these issues by applying a specific filtering step to (x',y'), bounding distances between facilities and clients they serve, as well retrieving the lost demand of every client in C'.

4.2 Filtering

We apply filtering to the restriction of (x',y') on $F' \times C'$, however, the filtering process will be quite different from [9] (the analysis though is similar). We will rely a lot on the fact that we now operate with non-λ-heavy facilities only.

▶ **Definition 18.** *Let $\rho := \frac{(1+\varepsilon)^2}{\varepsilon^2}$. For every $j \in C$ define $F'_j := \{i \in F' : d(i,j) \le \rho D(j)\}$.*

▶ **Lemma 19.** *For every $j \in C'$, $\sum_{i\in F'_j} x'_{ij} \ge 1/(\rho\varepsilon) = \varepsilon/(1+\varepsilon)^2$.*

Proof. Every $j \in C'$ was served by F' only, therefore $D(j) = \sum_{i \in F'} d(i,j)x'_{ij}$. Observe that at most a portion of $1/\rho$ demand of j can be served by facilities not in F'_j. Otherwise,

$$D(j) = \sum_{i \in F'} d(i,j)x'_{ij} \geq \sum_{i \in F' \setminus F'_j} d(i,j)x'_{ij} \geq \rho D(j) \sum_{i \in F' \setminus F'_j} x'_{ij} > \rho D(j) \cdot \frac{1}{\rho} = D(j),$$

a contradiction. Hence, $\sum_{i \in F' \setminus F'_j} x'_{ij} \leq 1/\rho$. Since $\sum_{i \in F'} x'_{ij} \geq \varepsilon/(1+\varepsilon)$ for all $j \in C'$, we have

$$\sum_{i \in F'_j} x'_{ij} = \sum_{i \in F'} x'_{ij} - \sum_{i \in F' \setminus F'_j} x'_{ij} \geq \frac{\varepsilon}{1+\varepsilon} - \frac{\varepsilon^2}{(1+\varepsilon)^2} = \frac{\varepsilon}{(1+\varepsilon)^2} = \frac{1}{\rho\varepsilon}.$$ ◀

We construct a new solution $(\hat{x}, \hat{y})$ as follows:

$$\text{for all } (i,j) \in F' \times C', \qquad \hat{x}_{ij} = \begin{cases} 0, & i \notin F'_j; \\ \frac{x'_{ij}}{\sum_{i \in F'_j} x'_{ij}}, & i \in F'_j; \end{cases} \qquad \hat{y}_i = \min\left(1, \rho\varepsilon y'_i\right).$$

Clearly, $\sum_{i \in F'} \hat{y}_i \leq \rho\varepsilon \sum_{i \in F'} y'_i \leq \rho\varepsilon k = O(k/\varepsilon)$. Also, by Lemma 19, $\hat{x}_{ij} \leq \min(1, \rho\varepsilon x'_{ij}) \leq \hat{y}_i$ for every $(i,j) \in F' \times C'$. To bound $L(i,\hat{x})$ for $i \in F'$, recall that i is non-λ-heavy, therefore $\sum_{j \in N'(i)} D(j) \leq \lambda T = \varepsilon^2 T/15$. Since $\hat{x}_{ij} > 0$ if and only if $x'_{ij} > 0$ and $d(i,j) \leq \rho D(j)$,

$$\lambda T \hat{y}_i \geq \sum_{j \in N'(i)} D(j)\hat{y}_i \geq \sum_{\substack{j \in N'(i) \\ s.t.\ \hat{x}_{ij} > 0}} D(j)\hat{y}_i \geq \frac{1}{\rho} \sum_{\substack{j \in N'(i) \\ s.t.\ \hat{x}_{ij} > 0}} d(i,j)\hat{y}_i \geq \frac{1}{\rho} \sum_{\substack{j \in N'(i) \\ s.t.\ \hat{x}_{ij} > 0}} d(i,j)\hat{x}_{ij},$$

implying

$$L(i,\hat{x}) = \sum_{\substack{j \in N'(i) \\ s.t.\ \hat{x}_{ij} > 0}} d(i,j)\hat{x}_{ij} \leq \rho\lambda T\hat{y}_i = \frac{\rho\varepsilon^2}{15} T\hat{y}_i = \frac{(1+\varepsilon)^2}{15} T\hat{y}_i.$$

Also, for every $j \in C'$ we now have $\sum_{i \in F'} \hat{x}_{ij} = 1$ and $\sum_{i : \hat{x}_{ij} > 0} \hat{y}_i \geq 1$.

Since $\{i \in F' : \hat{x}_{ij} > 0\} \subseteq N'(j)$ and $\{j \in C' : \hat{x}_{ij} > 0\} \subseteq N'(i)$, we will abuse the notation and redefine $N'(i)$ and $N'(j)$ in terms of assignment $\hat{x}$. Let $\hat{\nu} := \frac{(1+\varepsilon)^2}{15}$. It holds for $(\hat{x}, \hat{y})$:

1. $\sum_{i \in F'} \hat{y}_i \leq \rho\varepsilon k = O(k/\varepsilon)$,
2. for all $(i,j) \in F' \times C'$, $\hat{y}_i \geq \hat{x}_{ij}$ and if $\hat{x}_{ij} > 0$ then $d(i,j) \leq \rho D(j)$,
3. for every $j \in C'$, $\sum_{i \in F'} \hat{x}_{ij} = 1$ and $\sum_{i \in N'(j)} \hat{y}_i \geq 1$,
4. for every $i \in F'$, $L(i,\hat{x}) \leq \hat{\nu} T \hat{y}_i$.

4.3 Finishing the algorithm

Now we can correctly use our clustering and rerouting algorithms with $(\hat{x}, \hat{y})$. We subsequently apply Algorithm 2 and Algorithm 3 to $(\hat{x}, \hat{y})$ with newly defined sets N' for F' and C', with corresponding values of of parameters λ, ρ and $\nu \equiv \hat{\nu}$, obtaining the integral opening $\dot{y}$ and possibly fractional assignment $\dot{x}$ over (F', C'). By Lemma 14, for $h \in F'$ with $\dot{y}_h = 1$,

$$L(h,\dot{x}) \leq 3(\hat{\nu} + 4\rho\lambda)T = 3\left(\frac{(1+\varepsilon)^2}{15} + 4\frac{(1+\varepsilon)^2}{\varepsilon^2} \cdot \frac{\varepsilon^2}{15}\right) = (1+\varepsilon)^2 T \leq (1+3\varepsilon)T,$$

and we open at most $(1+\varepsilon)\sum_{i \in F'} \hat{y}_i = O(k/\varepsilon)$ facilities.

Since for every $j \in C$ we have $\sum_{i \in F} \dot{x}_{ij} = 1$, there is no need to modify fractional variables $\dot{x}$ to regain lost demand (as we loose nothing). Observe that all $i \in F \setminus F'$ serve $j \in C \setminus C'$ only, these j are assigned to $i \in F \setminus F'$ integrally, and for all $i \in F \setminus F'$ we have $L(i,\dot{x}) \leq (2+3\varepsilon)T$. Therefore, it remains to obtain integral assignment over (F', C'), where for every $i \in F'$

we have $L(i,\dot{x}) \le (1+3\varepsilon)T$. By applying minimum makespan rounding algorithm [7] to the restriction of $(\dot{x},\dot{y})$ on (F',C'), we get integral assignment, sacrificing a factor of 2 in load approximation for $i \in F'$, resulting in maximum load of the final solution at most $(2+6\varepsilon)T$. Algorithm 1 might have opened at most $O(k/\varepsilon^2)$ additional facilities, so we obtain a $\left(2+6\varepsilon, O(1/\varepsilon^2)\right)$-approximate solution to $\mathrm{SC}(T,k)$ problem, and the whole algorithm clearly runs in polynomial-time, proving Theorem 6.

References

1 Sara Ahmadian, Babak Behsaz, Zachary Friggstad, Amin Jorati, Mohammad R. Salavatipour, and Chaitanya Swamy. Approximation algorithms for minimum-load k-facility location. *ACM Transactions on Algorithms*, 14(2):16:1–16:29, 2018. URL: http://doi.org/10.1145/3173047.

2 Sara Ahmadian, Ashkan Norouzi-Fard, Ola Svensson, and Justin Ward. Better guarantees for k-means and euclidean k-median by primal-dual algorithms. In *Proceedings of the IEEE 58th Annual Symposium on Foundations of Computer Science*, FOCS 2017, pages 61–72, 2017. URL: http://doi.org/10.1109/FOCS.2017.15.

3 Esther M. Arkin, Refael Hassin, and Asaf Levin. Approximations for minimum and min-max vehicle routing problems. *Journal of Algorithms*, 59(1):1–18, 2006. doi:10.1016/j.jalgor.2005.01.007.

4 Jaroslaw Byrka, Thomas 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. URL: http://doi.org/10.1145/2981561.

5 Guy Even, Naveen Garg, Jochen Könemann, R. Ravi, and Amitabh Sinha. Covering graphs using trees and stars. In *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems*, APPROX 2003, 2003. doi:10.1007/978-3-540-45198-3_3.

6 M. Reza Khani and Mohammad R. Salavatipour. Improved approximation algorithms for the min-max tree cover and bounded tree cover problems. *Algorithmica*, 69:302–314, 2014. doi:10.1007/s00453-012-9740-5.

7 Jan Karel Lenstra, David B. Shmoys, and Éva Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1):259–271, 1990. doi:10.1007/BF01585745.

8 Shi Li. A 1.488 approximation algorithm for the uncapacitated facility location problem. In *Proceedings of the 38th International Colloquium on Automata, Languages and Programming*, ICALP 2011, pages 45–58, 2013. doi:10.1016/j.ic.2012.01.007.

9 Jyh-Han Lin and Jeffrey Scott Vitter. e-approximations with minimum packing constraint violation. In *Proceedings of the 24th Annual ACM Symposium on Theory of Computing*, STOC 1992, pages 771–782, 1992. doi:10.1145/129712.129787.

10 David B. Shmoys and Éva Tardos. Scheduling unrelated machines with costs. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA 1993, pages 448–454, 1993. URL: https://dl.acm.org/doi/10.5555/313559.313851.

A Preprocessing

▶ **Theorem 20** (Theorem 7 restated). *Let (x,y) be such that, for all $i \in F$, $L(i,x) \le \mu T y_i$ for some constant $\mu \ge 1$ and all other constraints of* SC-LP(T,k) *on variables x are satisfied. There exists a polynomial time algorithm that, given such solution (x,y) and a constant $\gamma \in (0,1)$, finds a solution (x',y') such that*

1. *$y' = y$, and if $x_{ij} = 0$, then $x'_{ij} = 0$;*
2. *for every $(i,j) \in F \times C$, $y'_i \ge x'_{ij}$, and if $x'_{ij} > 0$, then $x'_{ij} \ge \gamma y'_i$;*
3. *for every $j \in C$, $1 \ge \sum_{i \in F} x'_{ij} \ge 1-\gamma$;*
4. *for every $i \in F$, $L(i,x') \le (\mu + 2 - \gamma) T y'_i$.*

The algorithm we use in Theorem 7 is heavily inspired by the minimum makespan rounding algorithm, introduced by Lenstra et. al in [7]. In a sense, their algorithm achieves the desired property: in minimum makespan problem we have $y_i = 1$ for all $i \in F$, so for $j \in C$ we wish to have either $x'_{ij} = 0$ or $x'_{ij} = 1 = y'_i$. The key difference is that in our case y is *not* integral, which requires several modifications of the original algorithm.

Let (x, y) and $\gamma \in (0, 1)$ be given. Let $\tilde{F} \subseteq F$, $\tilde{C} \subseteq C$, let $E \subseteq \tilde{F} \times \tilde{C}$. Consider a bipartite graph $G = (\tilde{F} \cup \tilde{C}, E)$, let $\delta_E(v)$ be the neighbors of $v \in \tilde{F} \cup \tilde{C}$ in G, I.e., for $i \in \tilde{F}$, $\delta_E(i) = \{j \in \tilde{C} : (i, j) \in E\}$, and for $j \in \tilde{C}$, $\delta_E(j) = \{i \in \tilde{F} : (i, j) \in E\}$. For $(i, j) \in \tilde{F} \times \tilde{C}$ we introduce a variable w_{ij}, and numbers $d_j \leq 1$ and $L_i \leq \mu T y_i$, which can be thought of as the remaining demand of client $j \in \tilde{C}$ and the remaining load of facility $i \in \tilde{F}$ respectively. Given sets $\tilde{F}, \tilde{C}, E$ and numbers d, L, we define the polytope $P(\tilde{F}, \tilde{C}, E, d, L)$ as the solution set of the following feasibility linear program:

$$\begin{aligned}
&\sum_{i \in \delta_E(j)} w_{ij} = d_j, && \forall j \in \tilde{C}, \\
&\sum_{j \in \delta_E(i)} d(i,j) w_{ij} \leq L_i, && \forall i \in \tilde{F}, \\
&w_{ij} \leq \min(y_i, d_j), && \forall (i,j) \in E, \\
&w_{ij} \geq 0, && \forall (i,j) \in E.
\end{aligned} \tag{$P(\tilde{F}, \tilde{C}, E, d, L)$}$$

Note that all values y_i for $i \in F$ are fixed, so for every number d_j, $j \in C$, we have either constraint $\{w_{ij} \leq y_i\}$ or constraint $\{w_{ij} \leq d_j\}$. The extreme points of $P(\tilde{F}, \tilde{C}, E, d, L)$ possess some very important properties, which resemble the properties of the extreme point solutions to the auxiliary program for the minimum makespan rounding algorithm of [7].

▶ **Lemma 21.** *Let w be an extreme point of $P(\tilde{F}, \tilde{C}, E, d, L)$, where $d_j \geq \gamma$ for all $j \in \tilde{C}$. One of the following must hold:*

(a) *there exists $(i, j) \in E$ such that $w_{ij} = 0$,*
(b) *there exists $(i, j) \in E$ such that $w_{ij} = y_i$,*
(c) *there exists $(i, j) \in E$ such that $w_{ij} = d_j$,*
(d) *there eixsts $i \in \tilde{F}$ such that $|\delta_E(i)| \leq 1$,*
(e) *there exists $i \in \tilde{F}$ such that $|\delta_E(i)| = 2$ and $\sum_{j \in \delta_E(i)} w_{ij} \geq \gamma y_i$.*

Proof. Suppose that none of (a), (b), (c), or (d) hold. We will show that (e) must hold then.

For all $(i, j) \in E$ we have $0 < w_{ij} < \min(y_i, d_j)$, and for every $i \in \tilde{F}$ we have $|\delta_E(i)| \geq 2$. Since $\sum_{i \in \delta_E(j)} w_{ij} = d_j$ for all $j \in \tilde{C}$, we must also have $|\delta_E(j)| \geq 2$. As w is an extreme point of $P(\tilde{F}, \tilde{C}, E, d, L)$, there exist $\tilde{F}_* \subseteq \tilde{F}$ and $\tilde{C}_* \subseteq \tilde{C}$ such that $\sum_{i \in \delta_E(j)} w_{ij} = d_j$ for all $j \in \tilde{C}_*$, $\sum_{j \in \delta_E(i)} d(i, j) w_{ij} = L_i$ for all $i \in \tilde{F}_*$, $|\tilde{F}_*| + |\tilde{C}_*| = |E|$, and constraints corresponding to $\tilde{F}_*, \tilde{C}_*$ are linearly independent. Since $2|E| = 2|\tilde{F}_*| + 2|\tilde{C}_*| \leq \sum_{i \in \tilde{F}_*} |\delta_E(i)| + \sum_{j \in \tilde{C}_*} |\delta_E(j)| \leq 2|E|$, for all $i \in \tilde{F}_*$ we must have $|\delta_E(i)| = 2$, as well as $|\delta_E(j)| = 2$ for all $j \in \tilde{C}_*$. Therefore, the subgraph $G[\tilde{F}_* \cup \tilde{C}_*]$ of G induced on $\tilde{F}_* \cup \tilde{C}_*$ is a bipartite union of disjoint cycles.

Let H be a cycle of $G[\tilde{F}_* \cup \tilde{C}_*]$, let $H_{\tilde{F}_*} := H \cap \tilde{F}_*$, $H_{\tilde{C}_*} := H \cap \tilde{C}_*$. Since for all $i \in H_{\tilde{F}_*}$ we have $|\delta_E(i)| = 2$, $\delta_E(i) \subseteq H \cap E$, and similarly, as $|\delta_E(j)| = 2$ for all $j \in H_{\tilde{C}_*}$, $\delta_E(j) \subseteq H \cap E$. Suppose that (e) does not hold, then for all $i \in H_{\tilde{F}_*}$ we have $\sum_{j \in \delta_E(i)} w_{ij} < \gamma y_i$. Consequently,

$$\sum_{i \in H_{\tilde{F}_*}} y_i > \frac{1}{\gamma} \sum_{i \in H_{\tilde{F}_*}} \sum_{j \in \delta_E(i)} w_{ij} = \frac{1}{\gamma} \sum_{(i,j) \in H \cap E} w_{ij} = \frac{1}{\gamma} \sum_{j \in H_{\tilde{C}_*}} \sum_{i \in \delta_E(j)} w_{ij} = \frac{1}{\gamma} \sum_{j \in H_{\tilde{C}_*}} d_j.$$

The last equality follows from $\sum_{i \in \delta_E(j)} w_{ij} = d_j$ for every $j \in \tilde{C}_*$. Since $d_j \geq \gamma$ for all $j \in \tilde{C}$, $d_j \geq \gamma y_i$ for all $(i, j) \in E$. Since H is a cycle in bipartite graph, it has even length, its vertices alternate between $\tilde{F}_*$ and $\tilde{C}_*$, and $|H_{\tilde{F}_*}| = |H_{\tilde{C}_*}|$. Then, we can split vertices of H into disjoint

consecutive pairs (i,j), so that $i \in H_{\tilde{F}_*}$, $j \in H_{\tilde{C}_*}$, $(i,j) \in H \cap E$, and apply $d_j \geq \gamma y_i$ for every pair. Therefore, $\sum_{j \in H_{\tilde{C}_*}} d_j \geq \gamma \sum_{i \in H_{\tilde{F}_*}} y_i$, which combined with inequalities above leads to a contradiction. So, there must exist $i \in H_{\tilde{F}_*}$ such that $\sum_{j \in \delta_E(i)} w_{ij} \geq \gamma y_i$, implying (e). ◀

We transform (x,y) into (x',y') using a similar approach as in [7]. On every step $t \geq 1$ of the algorithm, we provide values of parameters $\tilde{F}^t, \tilde{C}^t, E^t, d^t, L^t$ so that polytope $P^t := P(\tilde{F}^t, \tilde{C}^t, E^t, d^t, L^t)$ is nonempty and $d^t_j \geq \gamma$ for $j \in \tilde{C}^t$, and find its extreme point w^t. By Lemma 21, either (a), (b), (c), (d) or (e) cases may occur for w^t. If (a), we set $x'_{ij} \leftarrow 0$, $E^{t+1} \leftarrow E^t \setminus \{(i,j)\}$. If (b), we set $x'_{ij} \leftarrow y_i$, $d^{t+1}_j \leftarrow d^t_j - y_i$, $L^{t+1}_i \leftarrow L^t_i - d(i,j)w^t_{ij}$, $E^{t+1} \leftarrow E^t \setminus \{(i,j)\}$. If (c), we set $x'_{ij} \leftarrow d^t_j$, $d^{t+1}_j \leftarrow 0$, $L^{t+1}_i \leftarrow L^t_i - d(i,j)w^t_{ij}$, $E^{t+1} \leftarrow E^t \setminus \{(i,j)\}$. If (d) or (e), we set $\tilde{F}^{t+1} \leftarrow \tilde{F}^t \setminus \{i\}$. After processing exactly one case (a), (b), (c), (d) or (e), we scan $j \in \tilde{C}^{t+1}$, and if $d^{t+1}_j < \gamma$ for some j, set $\tilde{C}^{t+1} \leftarrow \tilde{C}^{t+1} \setminus \{j\}$, $x'_{ij} \leftarrow 0$ for all (i,j) such that $i \in \delta_{E^{t+1}}(j)$, and then $E^{t+1} \leftarrow E^{t+1} \setminus \{(i,j) : i \in \delta_{E^{t+1}}(j)\}$. If the change of $\tilde{F}^{t+1}, \tilde{C}^{t+1}, E^{t+1}, d^{t+1}$ or L^{t+1} is not mentioned for current case, the values are as in step t, so even though we drop facility i from $\tilde{F}^t$ in case (d) or (e), the edges (i,j) for $j \in \delta_{E^t}(i)$ are still kept in E^{t+1}. Having processed $\tilde{C}^{t+1}$, if $E^{t+1} \neq \varnothing$, we move to step $t+1$ and consider P^{t+1}. Algorithm 4 gives the full pseudocode, summarizing all the steps.

Algorithm 4 Preprocessing.

Input: Initial values of $\tilde{F}$, $\tilde{C}$, E, d, L, parameter $\gamma \in (0,1)$.
Output: An assignment x'.

while $E \neq \varnothing$ **do**
  Find an extreme point w of $P(\tilde{F}, \tilde{C}, E, d, L)$
  if $\exists (i,j) \in E : w_{ij} = 0$ **then** $x'_{ij} \leftarrow 0$, $E \leftarrow E \setminus \{(i,j)\}$
  else if $\exists (i,j) \in E : w_{ij} = y_i$ **then**
    $x'_{ij} \leftarrow y_i$, $d_j \leftarrow d_j - y_i$, $L_i \leftarrow L_i - d(i,j)w_{ij}$, $E \leftarrow E \setminus \{(i,j)\}$
  else if $\exists (i,j) \in E : w_{ij} = d_j$ **then**
    $x'_{ij} \leftarrow d_j$, $d_j \leftarrow 0$, $L_i \leftarrow L_i - d(i,j)w_{ij}$, $E \leftarrow E \setminus \{(i,j)\}$
  else if $\exists i \in \tilde{F} : |\delta_E(i)| \leq 1$ **then** $\tilde{F} \leftarrow \tilde{F} \setminus \{i\}$
  else if $\exists i \in \tilde{F} : |\delta_E(i)| = 2$ **and** $\sum_{j \in \delta_E(i)} w_{ij} \geq \gamma y_i$ **then** $\tilde{F} \leftarrow \tilde{F} \setminus \{i\}$
  for $j \in \tilde{C}$ **do**
    if $d_j < \gamma$ **then**
      $\tilde{C} \leftarrow \tilde{C} \setminus \{j\}$, **for** $\boldsymbol{i \in \delta_E(j)}$ **do** $x'_{ij} \leftarrow 0$, $E \leftarrow E \setminus \{(i,j)\}$
return x', extended to $F \times C$ by adding zero entries

It is easy to see that if P^t is nonempty and $d^t_j \geq \gamma$ for $j \in \tilde{C}^t$, the very same holds for P^{t+1} in the next step, unless $E^{t+1} = \varnothing$. Indeed, we manually assure that for all j kept in $\tilde{C}^{t+1}$ the condition $d^{t+1}_j \geq \gamma$ must hold, and the restriction of w^t to the set $E^{t+1} \subseteq E^t$ is a feasible solution to P^{t+1}, by construction of the algorithm. Moreover, if we take $\tilde{F}^1 = F$, $\tilde{C}^1 = C$, $E^1 = \{(i,j) \in F \times C : x_{ij} > 0\}$, $d^1_j = 1$ for $j \in C$ and $L^1_i = \mu T y_i$ for $i \in F$, $d^1_j \geq \gamma$ and P^1 is nonempty, since there is a feasible solution $w_{ij} := x_{ij}$ for $(i,j) \in E^1$. We run Algorithm 4 with these initial values of $\tilde{F}$, $\tilde{C}$, E, d and L given as input, obtaining an assignment x'. By setting $y' := y$, we obtain a solution (x',y').

We claim that Algorithm 4 runs in polynomial-time, and solution (x',y') satisfies all requirements of Theorem 7. By Lemma 21, on every step $t \geq 1$ either (a), (b), (c), (d) or (e) must occur for w^t, the extreme point of P^t. Then, either $|E^t|$, $|\tilde{F}^t|$ or $|\tilde{C}^t|$ is reduced at least by 1 after step t. So, since $|E^1| \leq |F||C|$, after at most $2|F||C|$ steps we will have $E^{t+1} = \varnothing$ for some $1 \leq t \leq 2|F||C|$. Each step t takes only polynomial time to perform, thus the total running time is also polynomial.

Since $E^1 = \{(i,j) \in F \times C : x_{ij} > 0\}$, the only positive coordinates of x' can be (i,j) such that $x_{ij} > 0$, as if $x_{ij} = 0 \iff (i,j) \notin E^1$, Algorithm 4 sets $x'_{ij} = 0$ in the very end. The constraint $\{w_{ij} \le \min(y_i, d_j)\}$ of $P(\tilde{F}, \tilde{C}, E, d, L)$ assures that $x'_{ij} \le y'_i$, for all $(i,j) \in F \times C$. If $x'_{ij} > 0$, then either $x'_{ij} = y'_i$ (case (b)) or $x'_{ij} = d^t_j$ for some step $t \ge 1$ (case (c)). Since $y'_i \le 1$ and for all steps $t \ge 1$ we maintain $d^t_j \ge \gamma$ for all $j \in \tilde{C}$, in both cases we have $x'_{ij} \ge \gamma y'_i$.

Next, if after processing cases for w^t during some step $t \ge 1$ we end up with $d^{t+1} < \gamma$, client j gets discarded from $\tilde{C}^{t+1}$. Since $d^1_j = 1$ initially, by the end of step t we must have assigned at least $1-\gamma$ portion of j's demand before discarding j having $d^{t+1}_j < \gamma$. Then, after Algorithm 4 finishes, for all $j \in C$ we have $1 \ge \sum_{i \in F} x'_{ij} \ge 1 - \gamma$.

Finally, fix $i \in F$. Observe that if $i \in \tilde{F}^t$ in the beginning of step $t \ge 1$, then

$$\sum_{j \in \delta_{E^t}(i)} d(i,j) w^t_{ij} \le L^t_i = L^1_i - \sum_{C \setminus \tilde{C}^t} d(i,j) x'_{ij} \implies \sum_{C \setminus \tilde{C}^t} d(i,j) x'_{ij} + \sum_{j \in \delta_{E^t}(i)} d(i,j) w^t_{ij} \le \mu T y_i,$$

by feasibility of w^t for polytope P^t. Suppose that after step t facility i gets removed from $\tilde{F}^t$, so $i \notin \tilde{F}^{t+1}$. If case (d) occurred and $|\delta_{E^t}(i)| \le 1$, let $j \in \delta_{E^t}(i)$ be a single client served by facility i. After removing i from $\tilde{F}^t$, the constraint $\{\sum_{j \in \delta_E(i)} d(i,j) w_{ij} \le L_i\}$ is not present in P^{t+1} and all future-step polytopes. So, for any step $r \ge t+1$, the load we may get after obtaining w^r and determining the value of x'_{ij} is at most $d(i,j) w^r \le d(i,j) y_i \le T y_i$ (as $d(i,j) \le T$ for all $x_{ij} > 0$). The total load of facility i becomes $L(i,x') \le (\mu+1) T y_i$.

If case (e) occurred for this facility i, $|\delta_{E^t}(i)| = 2$ and $\sum_{j \in \delta_{E^t}(i)} w^t_{ij} \ge \gamma y_i$. Let j' and j'' be the two clients belonging to $\delta_{E^t}(i)$. Their contribution to facility i's load on step t is exactly $d(i,j') w^t_{ij'} + d(i,j'') w^t_{ij''}$, which is at most L^t_i. After removing i from $\tilde{F}^t$, the constraint $\{\sum_{j \in \delta_E(i)} d(i,j) w_{ij} \le L_i\}$ is not present in P^{t+1} and all future-step polytopes. So, for any step $r \ge t+1$, the load we may get after obtaining w^r and determining the values of both $x'_{ij'}$ and $x'_{ij''}$ is at most $d(i,j') w^r_{ij'} + d(i,j'') w^r_{ij''} \le d(i,j') y_i + d(i,j'') y_i$. Hence, the *additional* load facility i gained since the end of step t is at most

$$\begin{aligned}
&(d(i,j')y_i + d(i,j'')y_i) - (d(i,j')w^t_{ij'} + d(i,j'')w^t_{ij''}) = \\
&= d(i,j')(y_i - w^t_{ij'}) + d(i,j'')(y_i - w^t_{ij''}) \le T(2y_i - (w^t_{ij'} + w^t_{ij''})) \le \\
&\qquad \le T(2y_i - \gamma y_i) = (2-\gamma) T y_i.
\end{aligned}$$

Therefore, the total load of facility i becomes $L(i,x') \le (\mu + 2 - \gamma) T y_i$.

As a result, solution (x', y') and the preprocessing algorithm (Algorithm 4) indeed satisfy all the claimed properties of Theorem 7, thus finishing the proof.

B Hard instances

We first present a hard instance for MLkSC problem. Let R, M be integers, $R \ll M$. Let $k = 2R - 1$, $|F| = 2R$, $|C| = (M+R)R$. F and C are partitioned into R disjoint groups, each has exactly 2 facilities and exactly $M + R$ clients. For $i, h \in F$, $d(i,h) = 1$ if i, h are in the same group, otherwise $d(i,h) = R$. In every group, one facility has M collocated clients (call it M-facility), the other has R collocated clients (R-facility). The instance is illustrated in Figure 2.

There is a feasible fractional solution to MLkSC-LP for this instance with $T = 1$. Open every M-facility fully, and there assign all its collocated clients. Next, open every R-facility to $1 - 1/R$, and let it serve $(1 - 1/R)$-fraction of its collocated clients' demand. The remaining $1/R$ fraction of these clients' demand will be served by M-facility of the same group. It is easy to see that the load of every R-facility is 0, the load of every M-facility is $R \cdot 1/R \cdot 1 = 1$, and the opening is exactly $R \cdot (1 + 1 - 1/R) = 2R - 1 = k$.

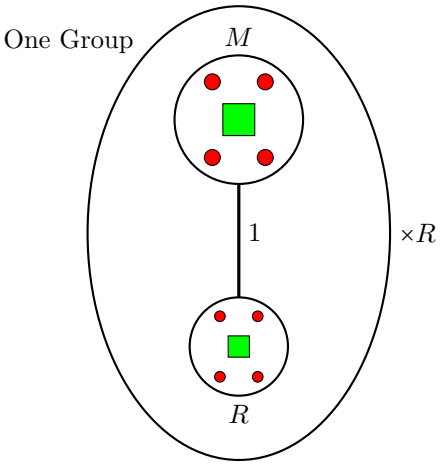


Figure 2 Hard instance for MLkSC-LP.

Consider any integral solution to this instance of MLkSC. If it assigns some client to a facility from different group, maximal load will be at least R. Suppose that all clients are assigned to facilities only from the same group. Since $k = 2R - 1$, there will be at least one group with at most one facility opened, take this group. If M-facility is opened, both its clients and clients of R-facility must be assigned to M-facility fully, resulting in its load $R \cdot 1 \cdot 1 = R$. Similarly, if R-facility is opened, maximum load will be at least $M \gg R$. Hence, the load of any integral star cover of size k is at least R. Furthermore, even if we allow opening $(1+\varepsilon)k$ facilities for $\varepsilon = 1/(2R)$, since

$$(1+\varepsilon)k = \left(1 + \frac{1}{2R}\right)(2R-1) = 2R - \frac{1}{2R} < 2R,$$

there will still be a group with at most one facility opened, resulting in maximum load at least $R = T/(2\varepsilon)$, where $T = 1$ is maximal fractional load. It follows that if T^* is an optimal load to MLkSC-LP, any integral $(1+\varepsilon)k$ star cover of (F, C) has load is at least $\Omega(1/\varepsilon)T^*$.

Now, we move to a hard instance for MSSC. For integer N, let $|F| = N$ and $|C| = N + 1$, the load bound $T \geq 1$ is arbitrary. Both $F = \{i_1, \ldots, i_N\}$ and $C = \{J, j_1, \ldots, j_N\}$ are vertices of a bipartite graph, and the metric d is a shortest-path metric. For every $1 \leq r \leq N$ we have an edge (i_r, j_r) or length $d(i_r, j_r) = (1 - 1/N)T$. Also, every facility i_r for $1 \leq r \leq N$ is connected to a "central" client J by an edge of length $d(i_r, J) = T$. The instance is illustrated in Figure 3.

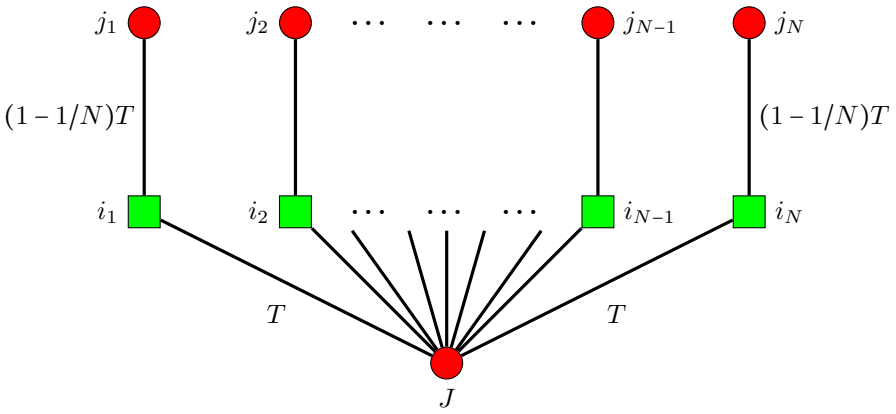


Figure 3 Hard instance for MSSC-LP.

It is easy to see that in any integral solution to MSSC-LP every client j_r for $1 \leq r \leq N$ can be served only by facility i_r. Furthermore, client J should also be served fully, so it should be assigned to one of $i \in F$. Therefore, even if we open all facilities in F fully, for some facility $i \in F$ which gets J assigned to it, the load will be at least $(2 - 1/N)T$. This means that there is no feasible integral solution to MSSC-LP, and any integral solution violates the maximum load constraint at least by a factor of $(2 - 1/N)$.

On the other hand, there exists a feasible *fractional* solution to MLkSC-LP for this instance. We open all i_r for $1 \leq r \leq N$ and assign j_r fully to it. Also, client J gets served by all $i \in F$ at an equal fraction of $1/N$. In this solution, the load of every facility $i \in F$ is exactly T.

On the Approximability of Presidential Type Predicates

Neng Huang
University of Chicago, IL, USA
nenghuang@uchicago.edu

Aaron Potechin
University of Chicago, IL, USA
potechin@uchicago.edu

Abstract

Given a predicate $P : \{-1,1\}^k \to \{-1,1\}$, let CSP($P$) be the set of constraint satisfaction problems whose constraints are of the form P. We say that P is approximable if given a nearly satisfiable instance of CSP(P), there exists a probabilistic polynomial time algorithm that does better than a random assignment. Otherwise, we say that P is approximation resistant.

In this paper, we analyze presidential type predicates, which are balanced linear threshold functions where all of the variables except the first variable (the president) have the same weight. We show that almost all presidential type predicates P are approximable. More precisely, we prove the following result: for any $\delta_0 > 0$, there exists a k_0 such that if $k \geq k_0$, $\delta \in (\delta_0, 1 - 2/k]$, and $\delta k + k - 1$ is an odd integer then the presidential type predicate $P(x) = \text{sign}(\delta k x_1 + \sum_{i=2}^{k} x_i)$ is approximable. To prove this, we construct a rounding scheme that makes use of biases and pairwise biases. We also give evidence that using pairwise biases is necessary for such rounding schemes.

2012 ACM Subject Classification Theory of computation → Rounding techniques

Keywords and phrases constraint satisfaction problems, approximation algorithms, presidential type predicates

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.58

Category APPROX

Related Version A full version of this paper is available at `https://arxiv.org/abs/1907.04451`.

1 Introduction

In constraint satisfaction problems (CSPs), we have a set of constraints and we want to satisfy as many of them as possible. Many fundamental problems in computer science are CSPs, including 3-SAT, MAX CUT, k-colorability, and unique games.

One fundamental question about CSPs is as follows. For a given type of CSP, is there a randomized polynomial time algorithm which is significantly better than randomly guessing an assignment? More precisely, letting r be the expected proportion of constraints satisfied by a random assignment, is there an $\epsilon > 0$ and a randomized polynomial time algorithm A such that given a CSP instance where at least $(1 - \epsilon)$ of the constraints can be satisfied, A returns an x which satisfies at least $(r + \epsilon)$ of the constraints in expectation? If so, we say that this type of CSP is approximable. If not, then we say that this type of CSP is approximation resistant.

For example, Håstad's 3-bit PCP theorem [10] proves that 3-XOR instances (where every constraint is a linear equation modulo 2 over 3 variables) are NP-hard to approximate. A direct corollary of Håstad's 3-bit PCP theorem is that 3-SAT is also NP-hard to approximate and this theorem has served as the basis for numerous other inapproximability results. On


© Neng Huang and Aaron Potechin;
licensed under Creative Commons License CC-BY



the other hand, Goemans and Williamson's [7] breakthrough algorithm for MAX CUT, which gives an approximation ratio of .878 for MAX CUT, shows that MAX CUT is approximable as a random cut would only cut half of the edges in expectation.

However, while the approximability or approximation resistance of CSPs has been extensively investigated, there is still much that is unknown. In this paper, we investigate CSPs where every constraint has the form of some fixed presidential type predicate P. We show that for almost all presidential type predicates P, this type of CSP is approximable.

1.1 Definitions

In order to better describe our results and their relationship to prior work, we need a few definitions.

▶ **Definition 1.** *A* Boolean predicate P *of arity* k *is a function* $P : \{-1,1\}^k \to \{-1,1\}$.

We remark that in general a predicate can be non-Boolean.

▶ **Definition 2.** *A* presidential type predicate *is a Boolean predicate of the form*

$$P(x_1, \ldots, x_k) = \text{sign}(a \cdot x_1 + x_2 + \cdots + x_k),$$

where a *is an integer and* $a + k - 1$ *is odd.*

▶ Remark 3. In the definition above we require that a is an integer. This is not a serious restriction because if a is not an integer, then we can shift a up or down slightly to find another predicate with integer coefficient a' which is equivalent to the original predicate. We require $a + k - 1$ to be odd in order to prevent a tie.

We can think of the predicate as a vote where the vote of x_1, the "president", has weight a, while the votes of the remaining voters, the "citizens", have the same weight 1.

▶ Remark 4. Note that presidential type predicates are balanced linear threshold functions, i.e. functions of the form $sign(\sum_{i=1}^k c_i x_i)$ where $\forall i, c_i \in \mathbb{R}$ and $\forall x \in \{-1,1\}^k, \sum_{i=1}^k c_i x_i \neq 0$ (so that the function is well-defined). Note that if a predicate P is a balanced linear threshold function, $P(-x) = -P(x)$ so exactly half of the assignments satisfy the predicate and thus a uniformly random assignment has expected value 0.

▶ **Definition 5.** *Given a Boolean predicate* $P : \{-1,1\}^k \to \{-1,1\}$*, an instance* Φ *of* $\text{CSP}(P)$ *consists of a set of* n *variables* $x_1, \ldots, x_n$ *and* m *constraints* $C_1, \ldots, C_m$ *where each* C_i *has the form*

$$C_i(x_{i_1}, \ldots, x_{i_k}) = P(z_{i,1}x_{i_1}, \ldots, z_{i,k}x_{i_k})$$

for some distinct $i_1, \ldots, i_k \in [n]$ *and* $z_{i,1} \ldots z_{i,k} \in \{-1,1\}$.

▶ **Definition 6.** *A Boolean predicate* P *is* approximable *if there exists a constant* $\epsilon > 0$ *and a polynomial time algorithm, possibly randomized, that on input* $\Phi \in \text{CSP}(P)$ *such that* $\text{OPT}(\Phi) \geq 1 - \epsilon$*, produces an assignment to* Φ*'s variables that in expectation satisfies* $r_P + \epsilon$ *fraction of the constraints in* Φ*, where* $r_P = \mathbb{E}_{x \in \{-1,1\}^k}[(1 + P(x))/2]$ *is the probability that a constraint in* Φ *is satisfied by a random assignment. Otherwise, we say* P *is* approximation resistant.

We say that a Boolean predicate P *is* weakly approximable *if there exists a constant* $\epsilon > 0$ *and a polynomial time algorithm, possibly randomized, that on input* $\Phi \in \text{CSP}(P)$ *such that* $\text{OPT}(\Phi) \geq 1 - \epsilon$*, produces an assignment to* Φ*'s variables that in expectation either satisfies at least* $r_P + \epsilon$ *fraction of the constraints in* Φ *or satisfies at most* $r_P - \epsilon$ *fraction of the constraints in* Φ*. Otherwise, we say that* P *is* strongly approximation resistant.

▶ Remark 7. For presidential type predicates, and in fact any odd predicate P (i.e. a predicate P where $P(-x) = -P(x)$), the notions of being approximable and being weakly approximable are equivalent.

1.2 Our Results

In this paper, we prove the following result.

▶ **Theorem 8.** *For any $\delta_0 > 0$, there exists a $k_0 \in \mathbb{N}$ such that if $k \geq k_0$, $\delta \in (\delta_0, 1 - 2/k]$, and $\delta k + k - 1$ is an odd integer then the presidential type predicate*

$$P(x) = \text{sign}\left(\delta k x_1 + \sum_{i=2}^{k} x_i\right)$$

is approximable.

▶ Remark 9. Informally, this theorem says that if the weight of x_1 (the "president") is at least a constant times k, then the predicate is approximable for sufficiently large k. We have the condition $\delta \leq 1 - 2/k$ because if $\delta > 1 - 2/k$ and $\delta k + k - 1$ is an odd integer, then $\delta k \geq k$, which means the predicate is a dictator predicate which is trivially approximable.

We will prove this theorem by constructing a *rounding scheme* that makes use of *biases* and *pairwise biases*, which are given by a standard semi-definite program (see Section 2.2). Complementarily, we also give evidence that using pairwise biases is necessary for such rounding schemes. In particular, we show that for any fixed $\delta > 0$ and degree m, for sufficiently large k there is no rounding scheme for the predicate $P(x) = \text{sign}\left(\delta k x_1 + \sum_{i=2}^{k} x_i\right)$ which has degree at most m and does not use pairwise biases (see Theorem 40).

1.3 Relationship to Prior Work

We now describe known criteria for determining whether a predicate P is approximable or approximation resistant and how our techniques compare to these criteria.

In 2008, Raghavendra [14] gave a characterization of which predicates are approximable and which predicates are approximation resistant. Raghavendra showed that either a standard semi-definite program (SDP) together with an appropriate rounding scheme gives a better approximation ratio than a random assignment or it is unique games hard to do so. However, this characterization leaves much to be desired because for a given predicate, it can be extremely hard to tell which case holds. In fact, it is not even known to be decidable!

Khot, Tulsiani, and Worah [12] gave a characterization of which predicates are weakly approximable which is based on whether there exist certain vanishing measures over a polytope which we call the KTW polytope (though similar polytopes were analyzed in some earlier papers, see e.g. [2, 3, 4, 11]). Unfortunately, it is also unknown whether this characterization is decidable.

Thus, if we want to determine if a given predicate P is approximable or approximation resistant, it is often better to use more direct criteria. For showing that predicates are hard to approximate, the following criterion, proved by Austrin and Mossel [5], is extremely useful.

▶ **Definition 10.** *We say that a Boolean predicate P has a balanced pairwise independent distribution of solutions if there exists a distribution D on $\{-1,1\}^k$ such that*

1. *D is supported on $\{x \in \{-1,1\}^k : P(x) = 1\}$ (D is a distribution of solutions to P)*
2. *For all $i \in [k]$, $\mathbb{E}_{x \in D}[x_i] = 0$ and for all $i < j \in [k]$, $\mathbb{E}_{x \in D}[x_i x_j] = 0$*

▶ **Theorem 11.** *If P has a balanced pairwise independent distribution of solutions then P is unique games hard to approximate.*

This criterion captures most but not all predicates which are known to be unique games hard to approximate. One example of a predicate which is not captured by this criterion is the predicate which was recently constructed by Potechin [13] which is unique games hard to approximate and is a balanced linear threshold function. [1]

For approximation resistance which does not rely on the hardness of unique games, Chan [6] gave the following stricter criterion which implies NP-hardness of approximation.

▶ **Theorem 12.** *If a predicate P has a balanced pairwise independent subgroup of solutions then P is NP-hard to approximate.*

For showing that predicates are approximable, the general technique is as follows:

1. Run Raghavendra's SDP to obtain biases $\{b_i : i \in [n]\}$ and pairwise biases $\{b_{ij} : i < j \in [n]\}$ for the variables.
2. Construct a rounding scheme which takes these biases and pairwise biases and gives us a solution x such that if the SDP "thinks" that almost all of the constraints are satisfiable then x satisfies significantly more constraints than a random assignment in expectation.

Based on rounding schemes which are essentially linear in the biases and pairwise biases, Hast [9] obtained the following criterion for when predicates are approximable:

▶ **Theorem 13** (Hast's criterion)**.** *Given a predicate $P : \{-1,+1\}^k \to \{-1,+1\}$,*

1. *Define $P_1 : \{-1,+1\}^k \to \mathbb{R}$ to be $P_1(x) = \sum_{i=1}^{k} \hat{P}_{\{i\}} x_i$*
2. *Define $P_2 : \{-1,+1\}^k \to \mathbb{R}$ to be $P_2(x) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \hat{P}_{\{i,j\}} x_i x_j$.*

If there are constants c_1, c_2 such that $c_2 \geq 0$ and $c_1 P_1(x) + c_2 P_2(x) > 0$ for all x such that $P(x) = 1$ then P is approximable.

Aside from Hast's criterion, most of the known approximability results are ad-hoc. Some such results are as follows.

1. Austrin, Benabbas, and Magen [1] showed that the monarchy predicate $P(x_1, \cdots, x_k) = \text{sign}((k-2)x_1 + \sum_{i=2}^{k} x_i)$ is approximable and that any predicate P which is a balanced symmetric quadratic threshold function is approximable.
2. Potechin [13] showed that the almost monarchy predicate $P(x_1, \cdots, x_k) = \text{sign}((k-4)x_1 + \sum_{i=2}^{k} x_i)$ is approximable for sufficiently large k.

In this paper, we prove that almost all presidential type predicates are approximable by generalizing the ideas Potechin [13] used to prove that the almost monarchy predicate is approximable for sufficiently large k and making these ideas more systematic. Our work compares to previous criteria as follows.

1. Raghavendra's criterion and the KTW criterion give a space of rounding schemes which should be considered but don't provide an efficient way to search for the best rounding scheme in this space. For our techniques, we take full advantage of this space of rounding schemes while also providing a way to systematically construct the rounding scheme which we need.

[1] There were previously known predicates, such as the GLST predicate [8] $P(x_1, x_2, x_3, x_4) = \frac{1+x_1}{2} x_2 x_3 + \frac{1-x_1}{2} x_2 x_4$, which are unique games hard (in fact NP-hard) to approximate yet do not have a balanced pairwise independent distribution of solutions. However, the hardness of these predicates can be reduced to the hardness of predicates which do have a balanced pairwise independent distribution of solutions, so Austrin and Mossel's criterion can still be used for these predicates.

2. Like Hast's criterion, we need to check that a certain expression is positive for all x such that $P(x) = 1$. However, there are two key differences between our techniques and Hast's criterion. First, as noted above, we use a larger space of rounding schemes. In particular, we use rounding schemes which are very much non-linear in the biases and pairwise biases. Second, because these rounding schemes are nonlinear in the biases and pairwise biases, it is not sufficient to check all x such that $P(x) = 1$. Instead, we need to check over the entire KTW polytope.

2 Techniques for Analyzing Boolean Predicates

In this section, we recall techniques for analyzing the approximability of boolean predicates.

2.1 Fourier Analysis

In this paper, we will make extensive use of the Fourier expansion of boolean predicates. The Fourier expansion of a k-ary boolean predicate P is of the form $P(x) = \sum_{I \subset [k]} \hat{P}_I x_I$, where $x_I = \prod_{i \in I} x_i$ and $\{\hat{P}_I : I \subseteq [k]\}$ are the Fourier coefficients $\hat{P}_I = \mathbb{E}_{x \in \{-1,1\}^k}[P(x)x_I]$ of P. We have the following lemma for the Fourier coefficients of presidential type predicates, the proof of which can be found in the full version.

▶ **Lemma 14** (Fourier coefficients of presidential type predicates). *Let $P(x_1, \ldots, x_k) = \mathrm{sign}(a \cdot x_1 + x_2 + \cdots + x_k)$ be a presidential type predicate where $a \leq k-2$ and $a + k - 1$ is an odd integer. Let $\hat{P}_{tC}$ denote the Fourier coefficient of a set of t citizens (indices from 2 to k) and $\hat{P}_{P+tC}$ denote the Fourier coefficient of a set of t citizens together with the president (index 1). Let $\tau = \lfloor (k-a-1)/2 \rfloor$. We have*

(1) $\hat{P}_P = 1 - \frac{1}{2^{k-2}} \sum_{l=0}^{\tau} \binom{k-1}{l}$,

(2) $\hat{P}_{tC} = \frac{1}{2^{k-2}} \sum_{i=0}^{\tau} \sum_{j=0}^{\tau-i} (-1)^j \binom{k-t-1}{i} \binom{t}{j}$, $\quad \forall t (1 \leq t \leq k-1 \wedge t \text{ is odd})$,

(3) $\hat{P}_{P+tC} = -\frac{1}{2^{k-2}} \sum_{i=0}^{\tau} \sum_{j=0}^{\tau-i} (-1)^j \binom{k-t-1}{i} \binom{t}{j}$, $\quad \forall t (2 \leq t \leq k-1 \wedge t \text{ is even})$.

2.2 Choosing Rounding Schemes

Our approximation algorithms for presidential type predicates work as follows. We first run the standard SDP given by Raghavendra [14]. This standard SDP gives us biases $\{b_i : i \in [n]\}$ and pairwise biases $\{b_{ij} : i < j \in [n]\}$ such that for each constraint, these biases and pariwise biases give us a point in the KTW polytope for that constraint, which is defined below and which plays a crucial role in Khot, Tulsiani and Worah's [12] characterization of which predicates are weakly approximable. For a more detailed discussion of this standard SDP and the KTW polytope, see the full version of this paper.

▶ **Definition 15.** *Given $x \in \{-1,1\}^k$, let $p(x) \in \{-1,1\}^{k+\binom{k}{2}}$ be the vector obtained by concatenating x and $(x_1x_2, x_1x_3, \ldots, x_{k-1}x_k)$. Define*

$$KTW_P = \mathrm{Conv}(\{p(x) \mid x \in \{-1,1\}^k, P(x) = 1\}),$$

where $P : \{-1,1\}^k \to \{-1,1\}$ is a boolean predicate and $\mathrm{Conv}(S)$ is the convex hull of S.

Once we have these biases $\{b_i : i \in [n]\}$ and pairwise biases $\{b_{ij} : i < j \in [n]\}$, we use a probabilistic rounding scheme to obtain an actual $x \in \{-1,1\}^n$. To choose this rounding scheme, we choose $\mathbb{E}[x_I]$ for each monomial $x_I = \prod_{i \in I} x_i$. However, we do not have complete freedom for these choices. Intuitively, $\mathbb{E}[x_I]$ should obey the following constraints:

1. $\mathbb{E}[x_I]$ is a function of $\{b_i \mid i \in I\}$ and $\{b_{ij} \mid i, j \in I\}$.
2. $\mathbb{E}[x_I]$ is invariant under permutations of the indices in I.
3. If we flip the sign of any variable x_i where $i \in I$ (by flipping the signs of b_i and $\{b_{ij} : j \in I, j \neq i\}$), then the sign of $\mathbb{E}[x_I]$ should be flipped as well.

It turns out that for determining whether a predicate P is *weakly approximable* (which is the same as approximable for presidential type predicates), these are the only constraints on $\mathbb{E}[x_I]$. More precisely, we have the following theorem from [13], which is also implicit in [12]:

▶ **Theorem 16** (Theorem 5.1 in [13]). *Let $\{b_i \mid i \in [k]\}$ and $\{b_{ij} \mid i, j \in [k], i < j\}$ be biases and pairwise biases produced by the standard SDP. For every $a \in [k]$, let $f_a : [-1,1]^{a+\binom{a}{2}} \to [-1,1]$ be a continuous function satisfying the following symmetric requirements.*

1. *For all permutations $\sigma \in S_a$,*

$$f_a(b_{i_{\sigma(1)}}, \ldots, b_{i_{\sigma(a)}}, b_{i_{\sigma(1)}i_{\sigma(2)}}, \ldots, b_{i_{\sigma(a-1)}i_{\sigma(a)}}) = f_a(b_{i_1}, \ldots, b_{i_a}, b_{i_1 i_2}, \ldots, b_{i_{a-1}i_a})$$

2. *For all signs $s_{i_1}, \ldots, s_{i_a} \in \{-1, 1\}^a$,*

$$f_a(s_{i_1}b_{i_1}, \ldots, s_{i_a}b_{i_a}, s_{i_1}s_{i_2}b_{i_1 i_2}, \ldots, s_{i_{a-1}}s_{i_a}b_{i_{a-1}i_a}) = f_a(b_{i_1}, \ldots, b_{i_a}, b_{i_1 i_2}, \ldots, b_{i_{a-1}i_a}) \cdot \prod_{j=1}^{a} s_{i_j}$$

Then there exists a sequence of rounding schemes $\{R_q\}$ and coefficients $\{c_q\}$ such that for all subsets $I = \{i_1, \ldots, i_a\}$ of size at most k,

$$\sum_q c_q \mathbb{E}_{R_q}[x_I] = f_a(b_{i_1}, \ldots, b_{i_a}, b_{i_1 i_2}, \ldots, b_{i_{a-1}i_a}),$$

where $\mathbb{E}_{R_q}[x_I]$ is the expected value of x_I given by rounding scheme R_q. Moreover, this sum can be taken to be globally convergent.

▶ Remark 17. This theorem gives us a linear combination of rounding schemes. The coefficients c_q can be thought of as a probability distribution of rounding schemes, but there are two problems:

1. $\sum_q |c_q|$ may not be 1. One fix to this issue is to scale f by an appropriate constant ϵ.
2. c_q may be negative. In general, this can be a real issue but here the predicates we consider are odd, which means if c_q is negative we can simply flip the rounding scheme R_q and take it with probability $-c_q$.

▶ **Example 18.** This theorem says the following about $\mathbb{E}[x_i]$ and $\mathbb{E}[x_i x_j]$.

1. We can take $\mathbb{E}[x_i] \sim f_1(b_i)$ for any continuous function f_1 such that $f_1(b_i) = -f_1(-b_i)$ (i.e. f_1 is odd).
2. We can take $\mathbb{E}[x_i x_j] \sim f_2(b_i, b_j, b_{ij})$ for any continuous function f_2 such that $f_2(b_i, b_j, b_{ij}) = f_2(b_j, b_i, b_{ij}) = -f_2(-b_i, b_j, -b_{ij})$. The first equality corresponds to exchanging i and j while the second equality corresponds to flipping x_i.

▶ **Example 19.** Some examples of possible functions f_3 are as follows:

1. We can take $\mathbb{E}[x_i x_j x_k] \sim x_i x_j x_k$
2. As discussed in the following subsections, we will take $\mathbb{E}[x_i x_j x_k] \sim (b_i b_{jk} + b_j b_{ik} + b_k b_{ij})$
3. Potechin [13] found a simpler rounding scheme for the monarchy predicate where $\mathbb{E}[x_i x_j x_k] \sim sign(x_i x_j x_k) \max\{|x_i|, |x_j|, |x_k|\}$

In choosing the rounding scheme, our goal is as follows. For each constraint, the standard SDP could give us any point in the KTW polytope. We need to show that no matter which point in the KTW polytope we are given, the probability that the rounding scheme satisfies the constraint is better than a random guess. Equivalently, we need to show that for all points in the KTW polytope, $\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I] > 0$.

▶ **Example 20.** Consider the majority predicate $P(x_1,\ldots,x_k) = \text{sign}(x_1+\ldots+x_k)$. If we take $E[x_i] = f_1(b_i) = \epsilon b_i$ and take $f_a = 0$ whenever $a > 1$ then

$$\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I] = \epsilon\hat{P}_{\{1\}}\sum_{i=1}^{k}b_i$$

Since $\sum_{i=1}^{k}x_i \geq 1$ for every satisfying assignment, for any point in the KTW polytope, $\sum_{i=1}^{k}b_i \geq 1$ and thus $\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I] \geq \epsilon\hat{P}_{\{1\}} > 0$.

3 Techniques for Approximating Presidential Type Predicates

In this section, we describe our techniques for approximating presidential type predicates. These techniques are a generalization of the techniques used in [13] to show that the almost monarchy predicate is approximable for sufficiently large k.

3.1 High Level Overview

To approximate the presidential type predicate $P(x) = \text{sign}\left(\delta k x_1 + \sum_{i=2}^{k}x_i\right)$, we use the following type of rounding scheme.

1. $f_1(b_i) = c_1 b_i$.
2. $f_{2l+1}(b_{i_1},\ldots,b_{i_{2l+1}},b_{i_1i_2},\ldots,b_{i_{2l}i_{2l+1}}) = c_{2l+1}\left(b_{i_1}b_{i_2i_3}\cdots b_{i_{2l}i_{2l+1}} + \text{symmetric terms}\right)$

where we need to carefully choose the coefficients $c_1, c_3, \ldots$ so that for all points in the KTW polytope,

$$\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I] > 0.$$

Because of the symmetry of presidential type predicates P, we can analyze $\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I]$ in terms of a few key functions of the biases and pairwise biases.

▶ **Definition 21.** *Given biases $\{b_i : i\in[k]\}$ and pairwise biases $\{b_{ij} : i<j\in[k]\}$, we make the following definitions:*

1. *We define $\alpha = b_1$*
2. *We define $\beta = \sum_{i=2}^{k}b_i$*
3. *We define $S_{\{\{i_1,i_2\}\}} = \sum_{1<i<j\in[k]}b_{ij}$. We then write $S_{\{\{i_1,i_2\}\}} = E(1+\Delta)$ where $E = \frac{\delta^2k^2}{2} - \frac{k}{2} + 1$ is the value we expect for $S_{\{\{i_1,i_2\}\}}$ and Δ measures how far $S_{\{\{i_1,i_2\}\}}$ is from this expected value.*

With these definitions, we can approximate $\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I]$ in terms of α, β, and Δ. Our strategy is now as follows:

1. We choose a polynomial $h(x) = \sum_{l=1}^{m}a_l x^l$ so that $h(1+\Delta)\approx 1$ except near $\Delta = -1$ as we must have that $h(0) = 0$. More precisely, we choose h to satisfy certain properties (see Lemma 36).

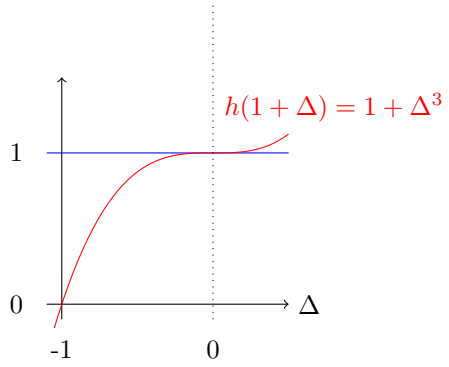


Figure 1 Plot of $h(1+\Delta) = 1+\Delta^3$.

▶ Remark 22. A reasonably good choice for h is $h(1+\Delta) = 1+\Delta^3$, which was used to give an approximation algorithm for the almost monarchy predicate for sufficiently large k [13]. In fact, while we don't prove it here, for quasi-monarchy predicates of the form $P(x_1,\ldots,x_k) = sign\left((k-2c)x_1 + \sum_{i=2}^{k} x_i\right)$ for a fixed constant c, $h(1+\Delta) = 1+\Delta^3$ is sufficient to give an approximation algorithm for sufficiently large k. However, this h is not sufficient to give an approximation algorithm for more general presidential type predicates because $h(1+\Delta) = 1+\Delta^3$ is far from 1 if Δ is much larger than 0.

2. We choose the coefficients $\{c_1\} \cup \{c_{2l+1} : l \in [m]\}$ so that

$$\begin{aligned}\sum_{I\subseteq[k]:I\neq\emptyset} \hat{P}_I \mathbb{E}[x_I] &= \left(\delta k^2 + \frac{k}{\delta}\right)\alpha + k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) + O(k)\cdot\Delta + O(1)\\ &= k(\delta k\alpha + \beta) + k\left(\beta - \frac{\alpha}{\delta}\right)(h(1+\Delta) - 1) + O(k)\cdot\Delta + O(1)\end{aligned}$$

3. Since for every satisfying assignment, $\delta k x_1 + \sum_{i=2}^{k} x_i \geq 1$, for every point in the KTW polytope,

$$\delta k b_1 + \sum_{i=2}^{k} b_i = \delta k\alpha + \beta \geq 1$$

and thus $k(\delta k\alpha + \beta) \geq k$. If we could show that the remaining terms $k\left(\beta - \frac{\alpha}{\delta}\right)(h(1+\Delta) - 1) + O(k)\cdot\Delta + O(1)$ are $o(k)$, then we would be done. Unfortunately, this may not be true when $|\Delta|$ is large.

4. To handle this, we show that if $|\Delta|$ is large then we can obtain a considerably better bound on $\delta k\alpha + \beta$ More precisely, we proceed as follows:
   a. When $\Delta \geq -0.55$, we show that $\delta k\alpha + \beta \geq \frac{(\delta^2 k-1)|\Delta|}{4} + \frac{1}{2}$ (see Lemma 37). As long as $h(1+\Delta)$ is sufficiently close to 1, this allows us to show that $k(\delta k\alpha + \beta) + k\left(\beta - \frac{\alpha}{\delta}\right)(h(1+\Delta) - 1) + O(k)\cdot\Delta + O(1)$ is positive.
   b. When $\Delta < -0.55$, we show that we must have $\alpha > 0$. In this case, we rewrite $\sum_{I\subseteq[k]:I\neq\emptyset} \hat{P}_I \mathbb{E}[x_I] = \left(\delta k^2 + \frac{k}{\delta}\right)\alpha + k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) + O(k)\cdot\Delta + O(1)$ as

$$k(\delta k\alpha + \beta)h(1+\Delta) + \left(\delta k^2 + \frac{k}{\delta}\right)\alpha(1 - h(1+\Delta)) + O(k)\cdot\Delta + O(1).$$

   and show that the sum of the first two terms is positive and $\Omega(k^2)$.

3.2 Sums of Products of Biases and Pairwise Biases

In order to implement this strategy, we need some notations related to biases and pairwise biases. Note that similar definitions were also used in [13].

▶ **Definition 23.** *For $E_1 \subseteq [k]$ and $E_2 \subseteq \binom{[k]}{2}$, define*

$$B_{E_1,E_2} = \prod_{i \in E_1} b_i \prod_{\substack{\{i,j\} \in E_2 \\ i<j}} b_{ij}.$$

▶ **Definition 24.** *Let $V = \{\alpha, i_1, i_2, \ldots, i_{k-1}\}$. Let $H = H_1 \cup H_2$ where $H_1 \subseteq V$ and $H_2 \subseteq \binom{V}{2}$. Define*

$$S_H = \sum_{\substack{E_1,E_2:\exists \sigma: V \to [k] \text{ bijective} \\ \sigma(\alpha)=1, \sigma(H_1)=E_1, \sigma(H_2)=E_2}} B_{E_1,E_2},$$

where $\sigma(H_1) = \{\sigma(i) \mid i \in H_1\}$, $\sigma(H_2) = \{\{\sigma(i), \sigma(j)\} \mid \{i,j\} \in H_2\}$.

Intuitively, S_H is the sum of products B_{E_1,E_2} where $E_1 \cup E_2$ has the form H. One particularly important such sum in our algorithms is $S_{\{\{i_1,i_2\}\}}$, which is the sum of pairwise biases with indices in $[2,k]$.

▶ **Definition 25.** *We define the following shorthand notations for some important sums.*

$$S_{1,l} = S_{\{i_1,\{i_2,i_3\},\{i_4,i_5\},\ldots,\{i_{2l},i_{2l+1}\}\}},$$
$$S_{2,l} = S_{\{\alpha,\{i_1,i_2\},\{i_3,i_4\},\ldots,\{i_{2l-1},i_{2l}\}\}},$$
$$S_{3,l} = S_{\{i_1,\{\alpha,i_2\},\{i_3,i_4\},\ldots,\{i_{2l-1},i_{2l}\}\}}.$$

▶ **Example 26.** In the case where $k = 4, l = 1$, we have

$$S_{1,l} = b_2b_{34} + b_3b_{24} + b_4b_{23},$$
$$S_{2,l} = b_1b_{23} + b_1b_{24} + b_1b_{34},$$
$$S_{3,l} = b_2b_{13} + b_2b_{14} + b_3b_{12} + b_3b_{14} + b_4b_{12} + b_4b_{13}.$$

The reason that these sums are important is because they are the main terms which appear when we evaluate $\sum_{I \subseteq [k]: I \neq \emptyset} \hat{P}_I \mathbb{E}[x_I]$.

▶ **Proposition 27.** *If we take*

$$f_{2l+1}(b_{i_1}, \ldots, b_{i_{2l+1}}, b_{i_1i_2}, \ldots, b_{i_{2l}i_{2l+1}}) = c_{2l+1}\left(b_{i_1}b_{i_2i_3}\cdots b_{i_{2l}i_{2l+1}} + \textit{symmetric terms}\right)$$

then

$$\sum_{|I|=2l+1} \hat{P}_I \mathbb{E}[X_I] = c_{2l+1}\left(\hat{P}_{(2l+1)^C} S_{1,l} + \hat{P}_{P+(2l)^C}(S_{2,l} + S_{3,l})\right).$$

To approximate these sums, we use the following proposition (recall that we set $S_{\{\{i_1,i_2\}\}} = E(1+\Delta)$). The proof of this proposition can be found in the appendix.

▶ **Proposition 28.** *For every $l \geq 1$,*

$$\frac{l!}{E^l} S_{1,l} = \beta(1+\Delta)^l - \frac{S_{\{i_1,\{i_1,i_2\}\}}}{E} l(1+\Delta)^{l-1} - \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2} l(l-1)(1+\Delta)^{l-2} + O\left(\frac{1}{k}\right),$$

$$\frac{l!}{E^l} S_{2,l} = \alpha(1+\Delta)^l + O\left(\frac{1}{k}\right),$$

$$\frac{l!}{E^l} S_{3,l} = \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E} l(1+\Delta)^{l-1} + O\left(\frac{1}{k}\right),$$

where the hidden constants in big-O may depend on l.

4 Proof of Theorem 8

In this section, we prove Theorem 8.

▶ **Theorem 8.** *For any $\delta_0 > 0$, there exists a $k_0 \in \mathbb{N}$ such that if $k \geq k_0$, $\delta \in (\delta_0, 1 - 2/k]$, and $\delta k + k - 1$ is an odd integer then the presidential type predicate*

$$P(x) = \operatorname{sign}\left(\delta k x_1 + \sum_{i=2}^{k} x_i\right)$$

is approximable.

In particular, we prove that for sufficiently large k and a carefully chosen polynomial h, the following rounding scheme approximates the presidential type predicate $P(x) = \operatorname{sign}\left(\delta k x_1 + \sum_{i=2}^{k} x_i\right)$.

▶ **Definition 29.** *Given a polynomial $h(x) = \sum_{l=1}^{m} a_l x^l$, we define $R_{k,\delta,h}$ to be the rounding scheme such that setting $u = \frac{1+\delta}{2}k$, $v = \frac{1-\delta}{2}k$, and $E = \frac{\delta^2 k^2}{2} - \frac{k}{2} + 1$,*

1. *$f_1(b_i) = \left(\delta k^2 + \frac{k}{\delta}\right) b_i$*
2. *For all $l \in [m]$,*

$$f_{2l+1}(b_{i_1}, \ldots, b_{i_{2l+1}}, b_{i_1 i_2}, \ldots, b_{i_{2l} i_{2l+1}}) = c_{2l+1}\left(b_{i_1} b_{i_2 i_3} \cdots b_{i_{2l} i_{2l+1}} + \textit{symmetric terms}\right)$$

where $c_{2l+1} = a_l \cdot \frac{2^{k-2}(u-1)!(v-1)!}{(k-2l-2)!\delta^{2l} k^{2l-1} E^l}$, .

▶ **Theorem 30.** *For all $\delta_0 > 0$, if $h = \sum_{l=1}^{m} a_l x^l$ is a polynomial such that*

1. *$h'(1) = h''(1) = 0$,*
2. *For all $\Delta \in [-0.55, \frac{1}{\delta_0^2}]$, $|h(1+\Delta) - 1| \leq \frac{\delta_0^2 |\Delta|}{5}$,*
3. *For all $\Delta \in [-1, -0.55]$, $0 \leq h(1+\Delta) \leq 1$,*

then there exists a $k_0 \in \mathbb{N}$ such that for all $\delta \geq \delta_0$ and $k \geq k_0$ where $\delta k + k - 1$ is an odd integer, $R_{k,\delta,h}$ approximates the presidential type predicate $P(x) = \operatorname{sign}\left(\delta k x_1 + \sum_{i=2}^{k} x_i\right)$.

▶ Remark 31. As described in Section 3.1, our proof contains a case analysis of Δ. The value -0.55 is chosen because when $\Delta < -0.55$, the bias α of the president is always positive.

This section is organized as follows. We first compute the expected value of the rounding scheme in terms of h. Then, we show that if h has the required properties, then the expected value is positive over the entire polytope, which implies that our predicate is approximable. Finally, we find such a polynomial with the desired properties.

4.1 Evaluating the Rounding Scheme

In this subsection, we analyze $\sum_{I\subseteq[k]:I\neq\emptyset}\hat{P}_I\mathbb{E}[x_I]$ in terms of h. We have the following lemma for the Fourier coefficients, the proof of which can be found in the full version.

▶ **Lemma 32.** *Let $\delta_0 > 0$ be a constant. Let $P(x_1,\dots,x_k) = \text{sign}(\delta\cdot kx_1 + x_2 + \cdots + x_k)$ where $\delta\in[\delta_0,1)$ such that $\delta k + k - 1$ is an odd integer. Let $u = \frac{1+\delta}{2}k$ and $v = \frac{1-\delta}{2}k$. Let $\hat{P}_{tC}$ denote the Fourier coefficient of a set of t citizens and $\hat{P}_{P+tC}$ denote the Fourier coefficient of a set of t citizens together with the president. We have the following:*

1. $\hat{P}_P = 1 - \frac{1}{2^{k-2}}\sum_{l=0}^{v-1}\binom{k-1}{l}$.
2. *If t is an odd integer, then*

$$\hat{P}_{tC} = \frac{1}{2^{k-2}}\cdot\frac{(k-t-1)!}{(u-1)!(v-1)!}\left(\delta^{t-1}k^{t-1} - \frac{(t-1)(t-2)}{2}\delta^{t-3}k^{t-2} + O(k^{t-3})\right).$$

3. *If t is an even integer, then*

$$\hat{P}_{P+tC} = -\frac{1}{2^{k-2}}\cdot\frac{(k-t-1)!}{(u-1)!(v-1)!}\left(\delta^{t-1}k^{t-1} - \frac{(t-1)(t-2)}{2}\delta^{t-3}k^{t-2} + O(k^{t-3})\right).$$

Here, the constants inside the big Os grows with t but not with δ.

▶ Remark 33. The lemma allows δ to depend on k as long as $\delta = \Omega(1)$. In particular, we can take $\delta = 1 - \frac{2c}{k}$ for any constant $c \geq 1$. Also, when δ is at least a constant we have that $\hat{P}_P$ is exponentially larger than $\hat{P}_C$.

Recall that we set $S_{\{\{i_1,i_2\}\}} = E(1+\Delta)$ where $E = \frac{\delta^2k^2}{2} - \frac{k}{2} + 1$ (see Definition 21). The reason for this choice for E is as follows. We expect that the cases which are most difficult to round are the two cases where $\delta k\alpha + b = 1$:

1. The president and $\frac{1-\delta}{2}k$ citizens vote 1, others vote -1. In this case,

$$\sum_{i<j\in[2,k]} x_ix_j = \binom{k-1}{2} - 2\frac{(1-\delta)k}{2}\left(\frac{(1+\delta)k}{2} - 1\right)$$

2. The president and $\frac{1+\delta}{2}k$ citizens vote -1, others vote 1. In this case,

$$\sum_{i<j\in[2,k]} x_ix_j = \binom{k-1}{2} - 2\frac{(1+\delta)k}{2}\left(\frac{(1-\delta)k}{2} - 1\right)$$

For both of these cases, $\sum_{i<j\in[2,k]} x_ix_j$ is approximately $\frac{\delta^2k^2}{2}$. Taking the average of these two cases we have $E = \frac{\delta^2k^2}{2} - \frac{k}{2} + 1$. Note that since $\delta > \delta_0$ is at least a constant, we have $E = \Omega(k^2)$.

The following lemma analyzes $\sum_{I\subseteq[k]:|I|\geq 3}\hat{P}_I\mathbb{E}[x_I]$ in terms of h. Its proof can be found in the appendix.

▶ **Lemma 34.** *Assume that we have $h(x) = \sum_{l=1}^{m} a_lx^l$ and coefficients*

$$c_{2l+1} = a_l\cdot\frac{2^{k-2}(u-1)!(v-1)!}{(k-2l-2)!\delta^{2l}k^{2l-1}E^l}$$

where $u = \frac{1+\delta}{2}k$, $v = \frac{1-\delta}{2}k$, and $E = \frac{\delta^2k^2}{2} - \frac{k}{2} + 1$. The contribution of degree ≥ 3 terms is

$$k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) - \frac{2(1+\Delta)^2\beta h''(1+\Delta)}{\delta^2} + \frac{(1+\Delta)\beta h'(1+\Delta)}{\delta^2}$$
$$- k\left(\frac{S_{\{i_1,\{i_1,i_2\}\}}}{E}h'(1+\Delta) + \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2}h''(1+\Delta) + \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E\delta}h'(1+\Delta)\right) + O(1)$$

▶ **Corollary 35.** *If we have* $h(x) = \sum_{l=1}^{m} a_l x^l$ *such that* $h'(1) = h''(1) = 0$ *and choose coefficients*

$$c_{2l+1} = a_l \cdot \frac{2^{k-2}(u-1)!(v-1)!}{(k-2l-2)!\delta^{2l}k^{2l-1}E^l}$$

then the contribution of degree ≥ 3 *terms is*

$$k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) + O(k)\cdot\Delta + O(1).$$

Proof. This follows from the fact that $E = \Omega(k^2), \beta = O(k), S_{\{i_1,\{i_1,i_2\}\}} = O(k^2)$, $S_{\{\{i_1,i_2\},\{i_1,i_3\}\}} = O(k^3)$ and $S_{\{\{\alpha,i_1\}\}} = O(k)$. ◀

4.2 Conditions on the Rounding Polynomial

▶ **Lemma 36.** *For all* $\delta_0 > 0$*, if* $h = \sum_{l=1}^{m} a_l x^l$ *is a polynomial such that*

1. $h'(1) = h''(1) = 0$
2. *For all* $\Delta \in [-0.55, \frac{1}{\delta_0^2}]$, $|h(1+\Delta) - 1| \leq \frac{\delta_0^2|\Delta|}{5}$
3. *For all* $\Delta \in [-1, -0.55]$, $0 \leq h(1+\Delta) \leq 1$

then there exists $k_0 \in \mathbb{N}$ *such that for all* $k \geq k_0$ *and all* $\delta \geq \delta_0$*, the rounding scheme* $R_{k,\delta,h}$ *has positive expected value over the entire KTW polytope.*

To prove this, we need the following lemma about points in the KTW polytope for P:

▶ **Lemma 37.** *For sufficiently large* k *we have*

$$\delta k\alpha + \beta \geq \frac{(\delta^2 k - 1)|\Delta|}{4} + \frac{1}{2}.$$

The proof of Lemma 37 can be found in the appendix.

Proof of Lemma 36. Since $h'(1) = h''(1) = 0$, by Corollary 35 the contribution of degree ≥ 3 terms becomes

$$k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) + O(k)\cdot\Delta + O(1).$$

Now we add in the contribution of degree 1 terms. Since $\hat{P}_P$ is extremely close to 1 and $\hat{P}_C$ is exponentially small, the contribution of degree 1 terms is extremely close to $c_1\alpha = (\delta k^2 + k/\delta)\alpha$. Adding this to the contribution from the higher degree terms, we get that

$$\sum_{I\subseteq[k]:I\neq\emptyset} \hat{P}_I \mathbb{E}[x_I] = k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) + \left(\delta k^2 + \frac{k}{\delta}\right)\alpha + O(k)\cdot\Delta + O(1). \qquad (*)$$

To establish the theorem, we need to show that $(*)$ is positive over the entire KTW polytope. We proceed with a case analysis on Δ. Note that the range of Δ is approximately $(-1 - O(1/k), 1/\delta^2)$. We have the following cases.

1. $\Delta \geq -0.55$. In this case we have

$$\begin{aligned}(*) &= k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) + \left(\delta k^2 + \frac{k}{\delta}\right)\alpha + O(k)\cdot\Delta + O(1)\\ &= k(\delta k\alpha + \beta) + k\left(\beta - \frac{\alpha}{\delta}\right)(h(1+\Delta) - 1) + O(k)\cdot\Delta + O(1)\\ &\geq k\left(\frac{(\delta^2 k - 1)|\Delta|}{4} + \frac{1}{2}\right) + k\left(\beta - \frac{\alpha}{\delta}\right)(h(1+\Delta) - 1) + O(k)\cdot\Delta + O(1)\end{aligned}$$

The last inequality is due to Lemma 37. Here the two terms which are quadratic in k are $\delta^2 k^2|\Delta|/4$ (note that $\delta > \delta_0$ is at least a constant) and $k\beta(h(1+\Delta)-1)$. Since $|\beta| < k$ and $|h(1+\Delta)-1| \leq \frac{\delta_0^2|\Delta|}{5}$, the above quantity is positive when k is sufficiently large.

2. $\Delta < -0.55$. Note that if $x_1 = -1$ then the minimum value of Δ is about 0. If $x_1 = 1$, then the minimum value of Δ is about -1. This means that when $\Delta < -0.55$, with probability > 0.5 we have $x_1 = 1$, which implies $\alpha > 0$ and is $\Omega(1)$. We can write $(*)$ as

$$(*) = k(\delta k\alpha + \beta)h(1+\Delta) + \left(\delta k^2 + \frac{k}{\delta}\right)\alpha(1 - h(1+\Delta)) + O(k)\cdot\Delta + O(1).$$

If $\Delta \geq -1$, then $h(1+\Delta) \in [0,1]$ and both the first two terms are positive and at least one of the two terms is $\Omega(k^2)$. If $\Delta \leq -1$ then since $\Delta \geq -1 - O(1/k)$ we know that the first term is $O(k)$ and the second term is $\Omega(k^2)$ and positive. Either way, we get a positive value when k is sufficiently large. ◀

4.3 Choosing the Rounding Polynomial

To finish the proof of our main theorem, we need to construct a polynomial that satisfies the conditions in Lemma 36. We claim that $h(x) = 1 - (1-x)^3\exp(-Bx)$ works for some constant B except that it's not a polynomial. However, by truncating the Taylor expansion of this function, we can get a polynomial which also works. See full version for the proof of these claims.

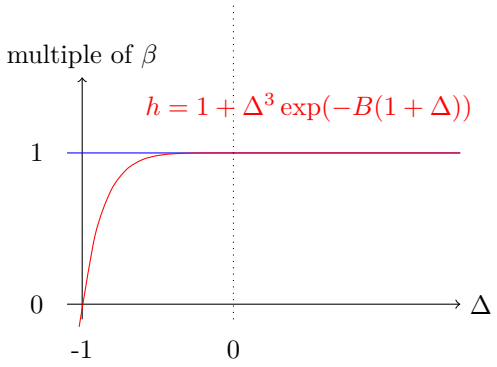


Figure 2 Plot of $h = 1 + \Delta^3\exp(-B(1+\Delta))$.

5 Evidence for the Necessity of Pairwise Biases

We have now given rounding schemes for almost all presidential predicates. These rounding schemes crucially use the pairwise biases $\{b_{ij} : i < j \in [k]\}$. A natural question is whether this is necessary or it is possible to only use the biases $\{b_i : i \in [k]\}$. If there is a rounding scheme which only uses the biases, then instead of using a semidefinite program, it is sufficient to use a linear program, which is much faster. Indeed, such rounding schemes exist for predicates which are close to the majority function [9] and for the monarchy predicate [5, 13].

In this section, we give evidence that this is not possible for more general presidential type predicates and it is necessary to use the pairwise biases. In particular, we prove the following theorem.

Recall that we choose a rounding scheme by specifying $f_a(b_{i_1}, \ldots, b_{i_a}, b_{i_1 i_2}, \ldots, b_{i_{a-1} i_a})$ for each $a \in [k]$.

▶ **Definition 38.** *We say that a rounding scheme has degree m if $f_m \neq 0$ and $f_a = 0$ for all $a > m$.*

▶ **Definition 39.** *We say that a rounding scheme does not use pairwise biases if for all $a \in [k]$, $f_a(b_{i_1}, \ldots, b_{i_a}, b_{i_1 i_2}, \ldots, b_{i_{a-1} i_a})$ only depends on $\{b_{i_1}, \ldots, b_{i_a}\}$.*

▶ **Theorem 40.** *For all $\delta_0 > 0$ and all $m \in \mathbb{N}$, there exists a k_0 such that for all $k \geq k_0$ and $\delta \in (\delta_0, 1 - 4/k]$ where $\delta k + k - 1$ is an odd integer, the presidential type predicate $P(x) = \operatorname{sign}\left(\delta k x_1 + \sum_{i=2}^{k} x_i\right)$ cannot be approximated by any rounding scheme of degree at most m which does not use pairwise biases.*

Proof. Let us consider a two-player zero-sum game where Alice chooses a point $b \in [-1, 1]^k$ in the KTW polytope[2] of P and Bob chooses a rounding scheme R of degree at most m. The objective of Alice is to minimize $R(b) = \sum_{I \subseteq [k]: I \neq \emptyset} \hat{P}_I \mathbb{E}[x_I]$, the expected value of $P(x)$ if we are given the point b in the KTW polytope and apply the rounding scheme R.

The lemma will follow if we can show a mixed strategy for Alice, which is a distribution μ over points in KTW polytope, such that for any rounding scheme R, $\mathbb{E}_{b \sim \mu}[R(b)] = 0$. Recalling that for each $a \in [m]$ and monomial $x_{i_1} x_{i_2} \ldots x_{i_a}$ of degree a, $\mathbb{E}[x_{i_1} x_{i_2} \ldots x_{i_a}] = f_a(b_{i_1}, \ldots, b_{i_a})$, it suffices to have the sum of degree a terms be zero for every $a \in [m]$, i.e.,

$$\mathbb{E}_{b \sim \mu} \left[\sum_{\substack{I \subset [k], |I| = a \\ I = \{i_1, \ldots, i_a\}}} \hat{P}_I f_a(b_{i_1}, \ldots, b_{i_a}) \right] = 0, \quad \forall a \in [m].$$

Now let us construct such a distribution μ. By Lemma 14, $\hat{P}_P$, the Fourier coefficient of the president x_1, is exponentially larger than $\hat{P}_C$, and $\lim_{k \to \infty} \hat{P}_{P+(t-1)C} / \hat{P}_{tC} = -1$ for every odd integer $t \leq m$. For concreteness, let us assume that $m = 5$. Then we will have the following distribution for μ:

Probability	x_1	x_2	x_3	x_4	x_5	x_6	x_7	$\cdots$	x_k
p_1	0	1	0	0	0	0	0	$\cdots$	0
p_2	0	1	1	-1	0	0	0	$\cdots$	0
p_3	0	1	1	1	0	0	0	$\cdots$	0
p_4	0	1	1	1	1	-1	0	$\cdots$	0
p_5	-1	1	1	1	1	1	1	$\cdots$	1

First of all, it is easy to check that all these points are inside the KTW polytope for $P(x)$. The following is a table of contribution of each degree from each of these points.

Degrees \ Points	1st type	2nd type	3rd type	4th type	5th type
$f_1(1)$	$p_1 \hat{P}_C$	$p_2 \hat{P}_C$	$3p_3 \hat{P}_C$	$3p_4 \hat{P}_C$	$p_5(-\hat{P}_P + (k-1)\hat{P}_C)$
$f_3(1,1,1)$	0	$-p_2 \hat{P}_{3C}$	$p_3 \hat{P}_{3C}$	$-2p_4 \hat{P}_{3C}$	$p_5(\binom{k-1}{2} \hat{P}_{P+2C} + \binom{k-1}{3} \hat{P}_{3C})$
$f_5(1,1,1,1,1)$	0	0	0	$-p_4 \hat{P}_{5C}$	$p_5(\binom{k-1}{4} \hat{P}_{P+4C} + \binom{k-1}{5} \hat{P}_{5C})$

To balance degree 1 terms, we need

$$p_1 \cdot \hat{P}_C + p_2 \cdot \hat{P}_C + p_3 \cdot 3\hat{P}_C + p_4 \cdot 3\hat{P}_C + p_5 \cdot (-\hat{P}_P + (k-1)\hat{P}_C) = 0.$$

[2] The KTW polytope of P actually has dimension $k + \binom{k}{2}$, but since pairwise biases play no role here, we omit those coordinates for simplicity.

Notice that every point in this distribution has a positive contribution from citizens (i.e., variables $x_2, \ldots, x_k$), so we need a negative contribution from x_1. Since $\hat{P}_P$ is exponentially larger than $\hat{P}_C$, we can achieve the balance by having p_5 be exponentially small in k. Then we balance degree 5 terms, for which we need

$$-p_4 \cdot \hat{P}_{5C} + p_5 \cdot \left(\binom{k-1}{4} \hat{P}_{P+4C} + \binom{k-1}{5} \hat{P}_{5C} \right) = 0.$$

Recall that $\lim_{k\to\infty} \hat{P}_{P+4C}/\hat{P}_{5C} = -1$, so we can achieve the balance by having $p_4 = poly(k) \cdot p_5$, where $poly(k)$ is a polynomial in k. For degree 3 terms, we need

$$-p_2 \cdot \hat{P}_{3C} + p_3 \cdot \hat{P}_{3C} - 2p_4 \cdot \hat{P}_{3C} + p_5 \cdot \left(\binom{k-1}{2} \hat{P}_{P+2C} + \binom{k-1}{3} \hat{P}_{3C} \right) = 0.$$

We can then use either the second type or the third type to balance degree 3 terms. Again we will only use $poly(k) \cdot p_5$ amount of probability. When k is sufficiently large, $p_2 + p_3 + p_4 + p_5 \leq 1$ and we let the first type of points take up the remaining probability. This method can be easily extended to handle the case where m is any fixed positive integer. ◀

▶ Remark 41. For the monarchy predicate $P(x) = sign\left((k-2)x_1 + \sum_{i=2}^{k} x_i\right)$, this argument fails for the following reason. The only satisfying assignment to the monarchy predicate where $x_1 = -1$ is when all of the other x_i are 1. This implies that for all $i \in [2, k]$, $b_i \geq -b_1$, which means that the point $b = (0, 1, 1, -1, 0, \ldots, 0)$ and similar points are not in the KTW polytope.

▶ Remark 42. This theorem rules out any fixed degree rounding schemes that use only biases, but it does not rule out the possibility that a rounding scheme might be able to succeed with just biases if its degree grows with k.

6 Conclusions

In this paper, we showed that almost all presidential type predicates are approximable. To do this, we carefully constructed rounding schemes which have positive expected value over the entire KTW polytope. These rounding schemes use both the biases $\{b_i : i \in [k]\}$ and the pairwise biases $\{b_{ij} : i < j \in [k]\}$ and have relatively high (but still constant) degree.

This work raises a number of open questions, including the following:

1. Which other types of predicates can this technique be applied to? For example, can we show that almost all oligarchy-type predicates are approximable, where oligarchy-type predicates are balanced LTFs where all but a few of the inputs have the same weight? As another example, can we extend the result of Austrin, Bennabas, and Magen that all symmetric quadratic threshold functions with no constant term are approximable to show that almost all quadratic threshold functions with no constant term which are symmetric with respect to all but one variable are approximable or at least weakly approximable?
2. Can we show that for almost all presidential type predicates, there is no rounding scheme which only uses the biases $\{b_i : i \in [k]\}$? Note that by Theorem 40, such rounding schemes would have to have degree which increases with k.
3. Our results only hold if k is sufficiently large. Is it true that all presidential type predicates are approximable? Less ambitiously, can we either extend our techniques or develop new techniques to handle presidential type predicates where k is relatively small?

References

1 Per Austrin, Siavosh Benabbas, and Avner Magen. On Quadratic Threshold CSPs. In Alejandro López-Ortiz, editor, *LATIN 2010: Theoretical Informatics*, Lecture Notes in Computer Science, pages 332–343. Springer Berlin Heidelberg, 2010.

2 Per Austrin and Johan Håstad. Randomly Supported Independence and Resistance. *SIAM Journal on Computing*, 40(1):1–27, January 2011. doi:10.1137/100783534.

3 Per Austrin and Johan Håstad. On the usefulness of predicates. *ACM Transactions on Computation Theory*, 5(1):1:1–1:24, May 2013. doi:10.1145/2462896.2462897.

4 Per Austrin and Subhash Khot. A characterization of approximation resistance for even k-partite CSPs. In *Proceedings of the 4th conference on Innovations in Theoretical Computer Science*, ITCS '13, pages 187–196, Berkeley, California, USA, January 2013. Association for Computing Machinery. doi:10.1145/2422436.2422459.

5 Per Austrin and Elchanan Mossel. Approximation Resistant Predicates from Pairwise Independence. *Computational Complexity*, 18(2):249–271, June 2009. doi:10.1007/s00037-009-0272-6.

6 Siu On Chan. Approximation Resistance from Pairwise-Independent Subgroups. *J. ACM*, 63(3):27:1–27:32, August 2016. doi:10.1145/2873054.

7 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.

8 Venkatesan Guruswami, Daniel Lewin, Madhu Sudan, and Luca Trevisan. A Tight Characterization of NP with 3-Query PCPs. In *Proc. of the 39th Symposium on Foundations of Computer Science*, pages 8–17. IEEE, 1998.

9 Gustav Hast. *Beating a Random Assignment : Approximating Constraint Satisfaction Problems*. PhD thesis, KTH Royal Institute of Tehnology, Sweden, 2005.

10 Johan Håstad. Some Optimal Inapproximability Results. *J. ACM*, 48(4):798–859, July 2001. doi:10.1145/502090.502098.

11 Howard Karloff and Uri Zwick. A 7/8-Approximation Algorithm for MAX 3SAT? In *Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, FOCS '97, page 406, USA, October 1997. IEEE Computer Society.

12 Subhash Khot, Madhur Tulsiani, and Pratik Worah. A Characterization of Approximation Resistance. *arXiv:1305.5500 [cs]*, May 2013. arXiv:1305.5500.

13 Aaron Potechin. On the Approximation Resistance of Balanced Linear Threshold Functions. *arXiv:1807.04421 [cs]*, July 2018. arXiv:1807.04421.

14 Prasad Raghavendra. Optimal Algorithms and Inapproximability Results for Every CSP? In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, pages 245–254, New York, NY, USA, 2008. ACM. doi:10.1145/1374376.1374414.

A Proof of Proposition 28

▶ **Proposition 28.** *For every $l \geq 1$,*

$$\frac{l!}{E^l} S_{1,l} = \beta(1+\Delta)^l - \frac{S_{\{i_1,\{i_1,i_2\}\}}}{E} l(1+\Delta)^{l-1} - \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2} l(l-1)(1+\Delta)^{l-2} + O\left(\frac{1}{k}\right),$$

$$\frac{l!}{E^l} S_{2,l} = \alpha(1+\Delta)^l + O\left(\frac{1}{k}\right),$$

$$\frac{l!}{E^l} S_{3,l} = \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E} l(1+\Delta)^{l-1} + O\left(\frac{1}{k}\right),$$

where the hidden constants in big-O may depend on l.

Proof. Here we only prove the first equality since the other two can be proved similarly. Recall that $S_{\{\{i_1,i_2\}\}} = E(1+\Delta)$ and $E = \Theta(k^2)$. The first equality is equivalent to

$$\begin{aligned} l!S_{1,l} =& \beta(S_{\{\{i_1,i_2\}\}})^l - S_{\{i_1,\{i_1,i_2\}\}} l (S_{\{\{i_1,i_2\}\}})^{l-1} - \beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}} l(l-1)(S_{\{\{i_1,i_2\}\}})^{l-2} \\ &+ O\left(k^{2l-1}\right). \end{aligned}$$

Let's analyze the term $\beta(S_{\{\{i_1,i_2\}\}})^l$, by definition, it's equal to

$$\left(\sum_{i\geq 2} b_i\right)\left(\sum_{2\leq i<j} b_{ij}\right)^l = \sum_{\substack{j_1,j_2,\ldots,j_{2l+1}\in\{2,3,\ldots,k\} \\ j_2<j_3, j_4<j_5,\cdots,j_{2l}<j_{2l+1}}} b_{j_1}b_{j_2j_3}b_{j_4j_5}\cdots b_{j_{2l}j_{2l+1}}.$$

Let's call the sum on the right hand side T. We classify the terms in T according to number of repetitions in indices. If there is no repetition, then the term $b_{j_1}b_{j_2j_3}b_{j_4j_5}\cdots b_{j_{2l}j_{2l+1}}$ is also in $S_{1,l}$. Note that in $S_{1,l}$ the order of the l pairwise biases can be arbitrary, so the sum of terms with no repeated indices is equal to $l!S_{1,l}$. If there are two or more repetitions, then the number of distinct indices is at most $2l-1$, and the contribution of such terms is $O(k^{2l-1})$. If there is exact one repetition, then there are two cases.

1. j_1 is equal to some j_t for $t \geq 2$. Without loss of generality consider the terms where the only repetition is $j_1 = j_2$ or $j_1 = j_3$ (note that $j_2 < j_3$). The contribution of these terms are

$$\begin{aligned} &\sum_{\substack{j_1,j_2,\ldots,j_{2l+1}\in\{2,3,\ldots,k\} \\ j_2<j_3, j_4<j_5,\cdots,j_{2l}<j_{2l+1} \\ j_1=j_2 \text{ or } j_1=j_3 \\ j_2,j_3,\ldots,j_{2l+1} \text{ distinct}}} b_{j_1}b_{j_2j_3}b_{j_4j_5}\cdots b_{j_{2l}j_{2l+1}} \\ =& \sum_{\substack{j_1,j_2,\ldots,j_{2l+1}\in\{2,3,\ldots,k\} \\ j_2<j_3, j_4<j_5,\cdots,j_{2l}<j_{2l+1} \\ j_1=j_2 \text{ or } j_1=j_3}} b_{j_1}b_{j_2j_3}b_{j_4j_5}\cdots b_{j_{2l}j_{2l+1}} + O(k^{2l-1}) \\ =& S_{\{i_1,\{i_1,i_2\}\}}(S_{\{\{i_1,i_2\}\}})^{l-1} + O(k^{2l-1}). \end{aligned}$$

   This is because the terms where $j_2, j_3, \ldots, j_{2l+1}$ are not distinct have at most $2l-1$ distinct indices and contribute $O(k^{2l-1})$. So the contribution of this case is

$$lS_{\{i_1,\{i_1,i_2\}\}}(S_{\{\{i_1,i_2\}\}})^{l-1} + O(k^{2l-1}).$$

2. $j_s = j_t$ for some $s,t \geq 2$. Note that in this case s and t cannot appear in the same pairwise bias. Without loss of generality assume $s \in \{2,3\}$ and $t \in \{4,5\}$. We have

$$\begin{aligned} &\sum_{\substack{j_1,j_2,\ldots,j_{2l+1}\in\{2,3,\ldots,k\} \\ j_2<j_3, j_4<j_5,\cdots,j_{2l}<j_{2l+1} \\ \text{one repetition in } j_2,j_3,j_4,j_5 \\ \text{other indices distinct}}} b_{j_1}b_{j_2j_3}b_{j_4j_5}\cdots b_{j_{2l}j_{2l+1}} \\ =& \sum_{\substack{j_1,j_2,\ldots,j_{2l+1}\in\{2,3,\ldots,k\} \\ j_2<j_3, j_4<j_5,\cdots,j_{2l}<j_{2l+1} \\ \text{one repetition in } j_2,j_3,j_4,j_5}} b_{j_1}b_{j_2j_3}b_{j_4j_5}\cdots b_{j_{2l}j_{2l+1}} + O(k^{2l-1}) \\ =& \quad 2\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}(S_{\{\{i_1,i_2\}\}})^{l-2} + O(k^{2l-1}). \end{aligned}$$

   So the contribution of this case is

$$\begin{aligned} &\binom{l}{2}\cdot\left(2\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}(S_{\{\{i_1,i_2\}\}})^{l-2} + O(k^{2l-1})\right) \\ =& \beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}} l(l-1)(S_{\{\{i_1,i_2\}\}})^{l-2} + O\left(k^{2l-1}\right). \end{aligned}$$

We conclude that

$$\beta(S_{\{\{i_1,i_2\}\}})^l = l!S_{1,l} + lS_{\{i_1,\{i_1,i_2\}\}}(S_{\{\{i_1,i_2\}\}})^{l-1} + \beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}} l(l-1)(S_{\{\{i_1,i_2\}\}})^{l-2} + O\left(k^{2l-1}\right).$$

We get the desired equality by shifting the terms. ◀

B Proof of Lemma 34

▶ **Lemma 34.** *Assume that we have* $h(x) = \sum_{l=1}^{m} a_l x^l$ *and coefficients*

$$c_{2l+1} = a_l \cdot \frac{2^{k-2}(u-1)!(v-1)!}{(k-2l-2)!\delta^{2l}k^{2l-1}E^l}$$

where $u = \frac{1+\delta}{2}k$, $v = \frac{1-\delta}{2}k$, *and* $E = \frac{\delta^2k^2}{2} - \frac{k}{2} + 1$. *The contribution of degree* ≥ 3 *terms is*

$$k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) - \frac{2(1+\Delta)^2\beta h''(1+\Delta)}{\delta^2} + \frac{(1+\Delta)\beta h'(1+\Delta)}{\delta^2}$$
$$- k\left(\frac{S_{\{i_1,\{i_1,i_2\}\}}}{E}h'(1+\Delta) + \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2}h''(1+\Delta) + \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E\delta}h'(1+\Delta)\right) + O(1)$$

Proof. We have the following computation:

$$\sum_{l=1}^{m}\sum_{|I|=2l+1}\hat{P}_I\mathbb{E}[X_I]$$
$$= \sum_{l=1}^{m} c_{2l+1}\left(\hat{P}_{(2l+1)C}S_{1,l} + \hat{P}_{P+(2l)C}(S_{2,l}+S_{3,l})\right)$$
$$= \sum_{l=1}^{m} a_l\Bigg(\left(k - l(2l-1)\delta^{-2} + O\left(\frac{1}{k}\right)\right)\cdot$$
$$\left(\beta(1+\Delta)^l - \frac{S_{\{i_1,\{i_1,i_2\}\}}}{E}l(1+\Delta)^{l-1} - \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2}l(l-1)(1+\Delta)^{l-2} + O\left(\frac{1}{k}\right)\right) -$$
$$\left(1 - \frac{2l+1}{k}\right)\cdot\left(\frac{k}{\delta} - \frac{(l-1)(2l-1)}{\delta^3} + O\left(\frac{1}{k}\right)\right)\cdot$$
$$\left(\alpha(1+\Delta)^l + \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E}l(1+\Delta)^{l-1} + O\left(\frac{1}{k}\right)\right)\Bigg)$$
$$= \sum_{l=1}^{m} a_l\Bigg(k\left(\beta - \frac{\alpha}{\delta}\right)(1+\Delta)^l - \frac{l(2l-1)\beta(1+\Delta)^l}{\delta^2}$$
$$- kl(1+\Delta)^{l-1}\left(\frac{S_{\{i_1,\{i_1,i_2\}\}}}{E} + \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2(1+\Delta)}(l-1) + \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E\delta}\right) + O(1)\Bigg)$$
$$= k\left(\beta - \frac{\alpha}{\delta}\right)h(1+\Delta) - \frac{2(1+\Delta)^2\beta h''(1+\Delta)}{\delta^2} + \frac{(1+\Delta)\beta h'(1+\Delta)}{\delta^2}$$
$$- k\left(\frac{S_{\{i_1,\{i_1,i_2\}\}}}{E}h'(1+\Delta) + \frac{\beta S_{\{\{i_1,i_2\},\{i_1,i_3\}\}}}{E^2}h''(1+\Delta) + \frac{\beta S_{\{\{\alpha,i_1\}\}}}{E\delta}h'(1+\Delta)\right) + O(1).$$ ◀

C Proof of Lemma 37

▶ **Lemma 37.** *For sufficiently large* k *we have*

$$\delta k\alpha + \beta \geq \frac{(\delta^2k-1)|\Delta|}{4} + \frac{1}{2}.$$

Proof. Since $|\Delta|$ is a convex function on the KTW polytope, it suffices to check that for each satisfying assignment, $\delta k\alpha + \beta \geq \frac{\delta k|\Delta|}{4} + \frac{1}{2}$. Letting t be the number of ones in $x_2, \ldots, x_k$, we have that $\beta = t - (k-1-t) = 2t - k + 1$ and

$$\sum_{2 \leq i < j} x_i x_j = \binom{t}{2} + \binom{k-1-t}{2} - t(k-1-t) = 2t^2 - 2(k-1)t + \binom{k-1}{2},$$

Recalling that $E = \frac{\delta^2 k^2}{2} - \frac{k}{2} + 1$, this implies that

$$\begin{aligned}\Delta = \frac{\sum_{2 \leq i < j} x_i x_j - E}{E} &= \frac{1}{E}\left(2t^2 - 2(k-1)t + \binom{k-1}{2} - \frac{\delta^2 k^2}{2} + \frac{k}{2} - 1\right) \\ &= \frac{1}{E}\left(2t^2 - 2(k-1)t + \frac{(1-\delta^2)k^2}{2} - k\right)\end{aligned}$$

Since $E > \frac{\delta^2 k^2}{2} - \frac{k}{2}$, we have

$$\begin{aligned}\frac{\delta^2 k - 1}{4}|\Delta| &= \frac{\delta^2 k - 1}{4E}\left|2t^2 - 2(k-1)t + \frac{(1-\delta^2)k^2}{2} - k\right| \\ &< \frac{1}{2k}\left|2t^2 - 2(k-1)t + \frac{(1-\delta^2)k^2}{2} - k\right| \\ &= \left|\frac{t^2}{k} - \frac{(k-1)t}{k} + \frac{(1-\delta^2)k}{4} - \frac{1}{2}\right|\end{aligned}$$

We will show that $\delta k\alpha + \beta \geq \left|\frac{t^2}{k} - \frac{(k-1)t}{k} + \frac{(1-\delta^2)k}{4} - \frac{1}{2}\right| + \frac{1}{2}$, from which our lemma will follow. To this end, we show that $\delta k\alpha + \beta \geq \left(\frac{t^2}{k} - \frac{(k-1)t}{k} + \frac{(1-\delta^2)k}{4} - \frac{1}{2}\right) + \frac{1}{2}$ and $\delta k\alpha + \beta \geq -\left(\frac{t^2}{k} - \frac{(k-1)t}{k} + \frac{(1-\delta^2)k}{4} - \frac{1}{2}\right) + \frac{1}{2}$.

1. $\delta k\alpha + \beta \geq \left(\frac{t^2}{k} - \frac{(k-1)t}{k} + \frac{(1-\delta^2)k}{4} - \frac{1}{2}\right) + \frac{1}{2}$.
   We have two cases, $\alpha = 1$ or $\alpha = -1$. If $\alpha = 1$, then $\delta k\alpha + \beta = \delta k + 2t - k + 1$ and since it's a satisfying assignment we have $t \geq \frac{1-\delta}{2}k$. The inequality becomes

   $$\frac{t^2}{k} - \frac{(k-1)t}{k} - 2t + \frac{(1-\delta^2)k}{4} - \delta k + k - 1 \leq 0.$$

   The left hand side is a quadratic function on t with positive leading coefficient, and to check it's non-positive we simply need to check its values on $t = \frac{1-\delta}{2}k$ and $t = k - 1$, the boundary points of t's domain. When $t = \frac{1-\delta}{2}k$,

   $$\begin{aligned}&\frac{t^2}{k} - \frac{(k-1)t}{k} - 2t + \frac{(1-\delta^2)k}{4} - \delta k + k - 1 = \\ &\left(\frac{t^2}{k} - t + \frac{(1-\delta^2)k}{4}\right) + (-2t - \delta k + k) + \left(\frac{t}{k} - 1\right) = \\ &\left(\frac{(1-\delta)^2 k}{4} - \frac{(1-\delta)k}{2} + \frac{(1-\delta^2)k}{4}\right) + 0 + \left(\frac{1-\delta}{2} - 1\right) = \\ &\frac{1-\delta}{2} - 1 < 0\end{aligned}$$

When $t = k - 1$,

$$\frac{t^2}{k} - \frac{(k-1)t}{k} - 2t + \frac{(1-\delta^2)k}{4} - \delta k + k - 1 =$$
$$\left(\frac{t^2}{k} - \frac{(k-1)t}{k}\right) + (-t + k - 1) + \left(-t - \delta k + \frac{(1-\delta^2)k}{4}\right) =$$
$$-\frac{3 - 4\delta + \delta^2}{4}k + 1$$

which is negative when k is sufficiently large (note that $\delta \leq 1 - \frac{3}{k}$).
If $\alpha = -1$, then $\delta k\alpha + \beta = -\delta k + 2t - k + 1$ and we have that $t \geq \frac{1+\delta}{2}k$. The inequality becomes

$$\frac{t^2}{k} - \frac{(k-1)t}{k} - 2t + \frac{(1-\delta^2)k}{4} + \delta k + k - 1 \leq 0.$$

We check the value of LHS on $t = \frac{1+\delta}{k}$ and $t = k-1$. Following exactly the same argument we used for $\alpha = 1$ except that δ is replaced by $-\delta$, when $t = \frac{1+\delta}{2}k$,

$$\frac{t^2}{k} - \frac{(k-1)t}{k} - 2t + \frac{(1-\delta^2)k}{4} + \delta k + k - 1 = \frac{1+\delta}{2} - 1 < 0$$

and when $t = k - 1$,

$$\frac{t^2}{k} - \frac{(k-1)t}{k} - 2t + \frac{(1-\delta^2)k}{4} + \delta k + k - 1 = -\frac{3 - 4\delta + \delta^2}{4}k + 1 < 0$$

2. $\delta k\alpha + \beta \geq -\left(\frac{t^2}{k} - \frac{(k-1)t}{k} + \frac{(1-\delta^2)k}{4} - \frac{1}{2}\right) + \frac{1}{2}$.
We again have two cases, $\alpha = 1$ or $\alpha = -1$. If $\alpha = 1$, we have $\delta k\alpha + \beta = \delta k + 2t - k + 1$ and the inequality becomes

$$\frac{t^2}{k} + \left(2 - \frac{k-1}{k}\right)t + \frac{(1-\delta^2)k}{4} + \delta k - k \geq 0.$$

The left hand side is a quadratic function that achieves minimum when t is negative, so we simply need the inequality to hold when $t = \frac{1-\delta}{2}k$, at which point the value of LHS is $\frac{1-\delta}{2} \geq 0$.
If $\alpha = -1$, the inequality becomes

$$\frac{t^2}{k} + \left(2 - \frac{k-1}{k}\right)t + \frac{(1-\delta^2)k}{4} - \delta k - k \geq 0.$$

Again, we simply need it to hold when $t = \frac{1+\delta}{2}k$, at which point the value of LHS is $\frac{1+\delta}{2} \geq 0$.
This completes our proof. ◀

An Approximation Algorithm for the MAX-2-Local Hamiltonian Problem

Sean Hallgren
Pennsylvania State University, State College, University Park, PA, USA
hallgren@cse.psu.edu

Eunou Lee
Pennsylvania State University, State College, University Park, PA, USA
eul153@psu.edu

Ojas Parekh
Sandia National Laboratories, Albuquerque, NM, USA
odparek@sandia.gov

Abstract

We present a classical approximation algorithm for the MAX-2-Local Hamiltonian problem. This is a maximization version of the QMA-complete 2-Local Hamiltonian problem in quantum computing, with the additional assumption that each local term is positive semidefinite. The MAX-2-Local Hamiltonian problem generalizes NP-hard constraint satisfaction problems, and our results may be viewed as generalizations of approximation approaches for the MAX-2-CSP problem. We work in the product state space and extend the framework of Goemans and Williamson for approximating MAX-2-CSPs. The key difference is that in the product state setting, a solution consists of a set of normalized 3-dimensional vectors rather than boolean numbers, and we leverage approximation results for rank-constrained Grothendieck inequalities. For MAX-2-Local Hamiltonian we achieve an approximation ratio of 0.328. This is the first example of an approximation algorithm beating the random quantum assignment ratio of 0.25 by a constant factor.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Semidefinite programming; Theory of computation → Quantum complexity theory

Keywords and phrases approximation algorithm, quantum computing, local Hamiltonian, mean-field theory, randomized rounding

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.59

Category APPROX

Funding *Sean Hallgren*: Partially supported by National Science Foundation awards CCF-1618287, CCF-1617710 and CNS-1617802, by a Vannevar Bush Faculty Fellowship from the US Department of Defense, and in part while the author was visiting the Simons Institute for the Theory of Computing. *Ojas Parekh*: Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia, LLC., a wholly owned subsidiary of Honeywell International, Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA-0003525. Supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Accelerated Research for Quantum Computing and Quantum Algorithms Teams programs.

Acknowledgements Thanks to Jianqiang Li and Sevag Gharibian for interesting discussions and useful comments.

1 Introduction

The k-Local Hamiltonian problem is the most studied QMA-complete problem in quantum computing [19] and generalizes classical constraint satisfaction problems (CSPs). It is physically motivated, asking about the ground state energy of a system specified by its Hamiltonian, or equivalently the minimum eigenvalue of an exponentially large Hermitian


© Sean Hallgren, Eunou Lee, and Ojas Parekh;
licensed under Creative Commons License CC-BY



matrix. Even though the matrix is exponentially large in the number of qubits, it can be succinctly described as a sum of matrices that are each a tensor product of a $2^k \times 2^k$ Hermitian matrix and the $2^{n-k} \times 2^{n-k}$ identity matrix, where n is the size of the whole matrix, and k is some constant given by a problem instance (called locality). Note that one only needs to describe the $2^k \times 2^k$ matrices to describe the whole problem. A precise definition will be given later. There are many variations depending on the locality, the number of levels each particle has (instead of 2 for the qubit case above), and other assumptions that can be made on the Hamiltonian. The class QMA has also been well-studied, and a large set of problems has been shown to be QMA-complete [6].

There has been much less progress in finding approximation algorithms for these problems. One difference arising in the quantum case is that classical algorithms cannot efficiently represent arbitrary solutions, which in general are entangled quantum states and thus reside in an exponentially large space (2^n dimensional). One approach to circumvent this problem is to only consider product state solutions, which are a subset of quantum states that exhibit no entanglement and admit efficient classical descriptions. Bansal, Bravyi, and Terhal [5] proved that a polynomial time approximation scheme (PTAS) (a polynomial-time algorithm that offers an arbitrarily good approximation at the expense of increase in run time) exists for Quantum Ising Spin Glass when assuming the underlying graph describing the interaction of the qubits is planar with bounded degree. Brandão and Harrow [7] analyze the D-regular case and give an additive approximation algorithm.

Another issue in the design of approximation algorithms is ensuring that problems have well-defined approximations. For general Hamiltonians, which are Hermitian, the spectrum can include positive and negative numbers, so it is difficult to define a meaningful approximation ratio. Gharibian and Kempe [12] defined the MAX-k-Local Hamiltonian problem and analyze the dense case. Their definition requires the k-local terms of the Hamiltonian to be positive semidefinite, so that meaningful approximations exist. This case is still QMA-hard and includes the classical constraint satisfaction problem, MAX-k-CSP as a special case.

The Max-2-Local Hamiltonian problem

In this paper we give a randomized classical approximation algorithm for the MAX-2-Local Hamiltonian (MAX-2-LH) problem on qubits. MAX-2-LH is QMA-hard, so approximation algorithms are a natural solution strategy. The problem is NP-hard because it includes MAX-2-CSP as a special case. To see that it is QMA-hard requires relating it to the decision (promise) k-Local Hamiltonian problem, defined on n qubits. In this problem, $2^k \times 2^k$ Hermitian matrices H_S are given for each subset $S \subseteq [n]$ of size k, together with real numbers a and b with $b - a \geq 1/poly(n)$. The H_S are called k-local terms and may be viewed as quantum generalizations of classical boolean constraints on k variables. The promise on the input is that either some eigenvalue of the Hamiltonian $H := \sum_S H_S \otimes I_{[n]\setminus S}$ is less than a, or all eigenvalues are greater than b (the notation $H_S \otimes I_{[n]\setminus S}$ signifies that H_S acts on the k qubits of S and is tensored with identity on the other $n - k$ qubits). The QMA witness for the yes case is a quantum state that is an eigenstate with eigenvalue less that a, and a quantum algorithm can verify in polynomial time whether its eigenvalue is less than a or larger by $1/poly(n)$. The QMA-hard 2-Local Hamiltonian problem may be reduced to the optimization problem, MAX-2-LH by negating H (since MAX-2-LH is a maximization problem) and adding identity terms, if necessary, to ensure that each 2-local term is positive semidefinite. MAX-2-LH also generalizes the NP-hard MAX-2-CSP problem, which is illustrated in Section 2.2.

For MAX-2-LH, we assume an edge set E over the vertex set $[n]$ and a set of 4×4 positive semidefinite Hermitian matrices $\{H_{pq}\}_{(p,q)\in E}$ are given. The goal is to compute the maximum eigenvector of $H := \sum_{(p,q)\in E} H_{pq} \otimes I_{[n]\setminus\{p,q\}}$. The Hermitian matrix H acts on n qubits, and each term H_{pq} acts on the two qubits p, q and as the identity on the remaining $n-2$ qubits. More precisely, $H_{pq} : \mathcal{H}_p \otimes \mathcal{H}_q \to \mathcal{H}_p \otimes \mathcal{H}_q$, where for all $p \in [n]$, $\mathcal{H}_p$ is the 2 dimensional complex Hilbert space, representing the qubit p, and $I_{[n]\setminus\{p,q\}}$ is the identity matrix on $\bigotimes_{i\in[n]\setminus\{p,q\}} \mathcal{H}_i$. The input matrices $\{H_{pq}\}_{(p,q)\in E}$ therefore implicitly describe the $2^n \times 2^n$ Hermitian matrix H, whose maximum eigenvector we wish to compute.

Related work

Other generalizations of classical problems to physically motivated Local Hamiltonian problems have been studied, but the exact relationship to well-studied classical problems such as MAX-2-SAT, MAX-CUT, and MAX-2-AND depends on which variant of the Local Hamiltonian problem is being considered. Gharibian and Parekh [14] study a maximization problem on 2-Local Hamiltonians for the Heisenberg model, which is a physically motivated generalization of MAX-CUT. These have the form $H = \sum_{(p,q)\in E} w_{pq} H_{pq}$ for $w_{pq} \geq 0$, where $H_{pq} = I - \alpha X_p X_q - \beta Y_p Y_q - \gamma Z_p Z_q$, for $\alpha, \beta, \gamma \in \{0, 1\}$, where X_p, Y_p, Z_p are the Pauli matrices on qubit p (e.g., X_p is a $2^n \times 2^n$ matrix formed by taking the 2×2 Pauli X matrix on qubit p tensored with identity on all the other $n-1$ qubits). They get approximation ratios 0.498 (when $\alpha+\beta+\gamma = 3$), 0.649 (when $\alpha+\beta+\gamma = 2$), or 0.878 (when $\alpha+\beta+\gamma = 1$). Furthermore, they show that their ratios are almost tight in the product state space. In particular they give a simple instance on which a product state cannot provide an approximation ratio better than 0.5, for the $\alpha+\beta+\gamma = 3$ case. This case of their problem is a special case of MAX-2-LH, hence 0.5 is also an upper bound on the best attainable approximation ratio for MAX-2-LH using product states. Anshu, Gosset, and Morenz [2] have very recently demonstrated that it is possible for a classical algorithm to provide an approximation ratio better than 0.5 for this problem, by outputting descriptions of relatively simple entangled states.

Bravyi, Gosset, König, and Temme [8] consider traceless 2-local Hamiltonians. This generalizes the Maximum Quadratic Programming problem (MAX-QP) where the diagonal entries are zero [10, 3]. In addition to generalizing MAX-QP, it also captures a variety of physical quantum models. They give a randomized classical algorithm that outputs a product state with expected energy $\mathrm{OPT}/O(\log n)$, where OPT is the maximum eigenvalue of the input Hamiltonian.

Our results

In this paper we achieve an approximation ratio of 0.328 for MAX-2-LH. The approximation algorithm for MAX-2-LH that picks a random assignment (i.e., the fully mixed quantum state) achieves ratio 0.25. Ours is the first example of an approximation algorithm for this problem that beats the trivial bound by a constant. It is also possible to achieve a ratio of $0.25 + 1/O(\log(n))$ by taking the input instance, shifting each term H_i so that it is traceless, and using the result in [8]. In MAX-2-LH each local term, acting on 2 qubits, is a rank 1 projector. In the special case when each local term is a rank 1 projector that is a product of two rank 1 projectors, one on each qubit, we get a better ratio of 0.40. This special case is detailed in Section 4. It is QMA-hard to find the maximum eigenvalue of this special type of Hamiltonian by Theorem 7 in [11].

In terms of techniques, we follow the framework that was first introduced by Goemans and Williamson. We first formulate a semidefinite program (SDP) whose optimal objective value provides an upper bound on the maximum energy of a given 2-local Hamiltonian. We can solve the SDP to arbitrary precision in polynomial time. Then we randomly round the solution, which resides in a larger space, down to the original solution space. Analyzing the randomized rounding is considerably more complicated than in the classical setting, because we need to round the solutions to a continuous 3-dimensional space, whereas the solution space is the boolean space for classical cases. We build upon a rounding procedure analyzed for generalizing the positive-semidefinite Grothendieck problem to a rank-constrained setting, by Briët, de Oliveira Filho, and Vallentin [9].

Even with this rounding procedure, obtaining a multiplicative bound that outperforms a random assignment for MAX-2-LH is a nontrivial task. In particular, our analysis takes the 1-local part of the input Hamiltonian explicitly into account. Ours is the first work of its kind to do so. Bravyi, Gosset, König, and Temme [8] showed that in the context of their problem, 1-local terms could be ignored without loss of generality, while Gharibian and Parekh [14] considered a problem without 1-local terms.

It is natural to consider SDP relaxations for our problem, because most approximation algorithms for classical MAX-2-CSP employ SDP rounding [15, 23, 22, 20]. Our SDP formulation is similar to that of [14, 8]; however, we must use additional constraints to handle the 1-local part. We also consider a different SDP formulation in order to get an improved approximation ratio in the special case where each local term is a product projector. The alternative formulation is described in Section 4. An advantage of the first SDP formulation is that the optimal value of the program provides an upper bound on OPT itself (the maximum eigenvalue of the Hamiltonian), whereas the optimal value of the alternative program gives an upper bound on $\mathrm{OPT}_{\mathrm{prod}}$, the objective value achieved by the best product state.

Open questions

There are several open questions. Is it possible to close the gap between 0.328 that we achieve and the 0.5 upper bound for product states? Is it possible to beat the uniformly random assignment when each term is a rank 3 projector, which is a quantum generalization of the classical Max 2-SAT problem? Is it possible to use more general states than product states, for example, as in the recent developments of Anshu, Gosset, and Morenz [2]? Does the approximation algorithm shed any light on the quantum PCP theorem?

2 Background

2.1 Approximating quantum problems

We first define the MAX-2-Local Hamiltonian problem.

▶ **Definition 1** (MAX-2LH). *An instance is given as a set of Hamiltonians* $\{H_{pq} : (p,q) \in E\}$, *for some edge set* $E \subseteq [n] \times [n]$, *where* $H_{pq} \otimes I_{[n]\setminus\{p,q\}}$ *is positive semidefinite and operates non-trivially on qubits* p *and* q. *Given such a list of Hamiltonians on 2 qubits, the goal of the problem is to find the largest eigenvalue of* $H = \sum_{(p,q)\in E} H_{pq} \otimes I_{[n]\setminus\{p,q\}}$, *which we denote as* OPT.

To simplify notation we will write $H = \sum_{pq} H_{pq}$, *where terms* H_{pq} *for* $(p,q) \notin E$ *are the zero matrix (so have rank 0). Also, let* $\mathrm{OPT}_{\mathrm{prod}}$ *denote the maximum energy achievable over the set of product states, i.e.,* $\mathrm{OPT}_{\mathrm{prod}} = \max_{|\phi_1\rangle\cdots|\phi_n\rangle} \langle\phi_1| \cdots \langle\phi_n| H |\phi_1\rangle \cdots |\phi_n\rangle$.

Since the 2-Local Hamiltonian problem can be reduced to the MAX-2-Local Hamiltonian problem, this problem is QMA-hard. So instead, we turn to approximating OPT with a multiplicative error. The goal of this paper is to beat the trivial approximation for the maximum eigenvalue of 2-local Hamiltonians presented in the following theorem.

▶ **Theorem 2.** *Given a 2-local Hamiltonian* $H = \sum_{pq} H_{pq} \otimes I_{[n]\setminus\{p,q\}}$ *with local terms that are rank* r *projectors,* H_{pq} *(*$r \in \{1,2,3\}$*), if the maximum eigenvalue (energy) of* H *is* OPT*, then the uniformly random product state achieves energy at least* $r/4 \cdot \mathrm{OPT}$*.*

Proof. Consider the contribution of the local term H_{pq} to the energy. Since we are assigning the uniformly random product state, the qubit p, q is assigned

$$\rho_{pq} := \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} \otimes \begin{pmatrix} 1/2 & 0 \\ 0 & 1/2 \end{pmatrix} = I/4$$

jointly. The energy contribution of H_{pq} is then $\mathrm{Tr}(H_{pq}\rho_{pq}) = \mathrm{Tr}(H_{pq})/4 = r/4$, because H_{pq} is a rank r projector. Therefore if ρ_n is the uniformly random product state on n qubits, the total energy is $\mathrm{Tr}(H\rho_n) = rm/4$ where m is the number of local terms. We know that $m \geq \mathrm{OPT}$. So $\mathrm{Tr}(H\rho_n) \geq r/4 \cdot \mathrm{OPT}$. ◀

Harrow and Montanaro [16] consider approximation of the minimum and maximum eigenvalue of local Hamiltonians where each qubit can appear in at most D local terms. They give an algorithm with approximation ratio $c + \Omega(1/\sqrt{D})$, where c is the ratio achieved by uniformly random assignment. More related to the problem we consider, Gharibian and Kempe [12] consider approximating the maximum eigenvalue of a k-local Hamiltonian with positive semidefinite local terms.

▶ **Theorem 3** ([12])**.** *For a* k*-local Hamiltonian* H *on* n*,* d*-dimensional qudits with positive semidefinite local terms and maximum eigenvalue (energy)* OPT*, there exists a product state assignment with energy of* OPT/d^{k-1}*.*

We are interested in the case of $(d, k) = (2, 2)$ in this paper. In this case, the above theorem implies there exists a product state with energy $\mathrm{OPT}/2$ and this means $\mathrm{OPT}_{\mathrm{prod}} \geq \mathrm{OPT}/2$; however, finding such a product state is NP-hard in general. They also give an approximation algorithm achieving a constant approximation ratio for dense instances.

▶ **Theorem 4** ([12])**.** *For any* k*-local Hamiltonian* H *on* n *qubits, there exists a deterministic poly-time algorithm outputting a product state achieving* $\langle\psi|H|\psi\rangle > \mathrm{OPT}_{\mathrm{prod}} - \epsilon n^k$*.* $\mathrm{OPT}_{\mathrm{prod}}$ *is the highest energy that can be achieved by a product state.*

In the case of $k = 2$, this implies a product state with energy $\mathrm{OPT}_{\mathrm{prod}} - \epsilon n^2$ can be efficiently found. If the constraint graph is dense, combining the two gives a $(1/2 - \epsilon')$-approximation algorithm. With a dense constraint graph, we know that $\mathrm{OPT}_{\mathrm{prod}} = \Theta(n^2)$ (by assigning a random product state). So $\mathrm{OPT}_{\mathrm{prod}} - \epsilon n^2 = \mathrm{OPT}_{\mathrm{prod}} - c\epsilon\,\mathrm{OPT}_{\mathrm{prod}} = (1 - c\epsilon)\,\mathrm{OPT}_{\mathrm{prod}}$, for some constant c. By Theorem 3, we know that this value is at least $(1/2 - c\epsilon/2)\,\mathrm{OPT}$.

Brandão and Harrow's result [7] also says that product states do better on regular constraint graphs as the degree increases.

▶ **Theorem 5** ([7])**.** *Suppose a 2-local Hamiltonian* H *on qudits has non-trivial terms on a* D*-regular graph. For all edges* (p, q) *on the graph, suppose the local term* H_{pq} *satisfies* $\|H_{pq}\| \leq 1$*. Then there exists a product state* $|\psi\rangle$ *such that* $\langle\psi|H|\psi\rangle \geq \mathrm{OPT} - 12\frac{nD}{2}(\frac{d^2 \ln d}{D})^{1/3}$*, where* OPT *is the largest eigenvalue of* H*.*

The above theorems imply that there exists a product state that achieves approximation ratio of $1 - 12(\frac{d^2 \ln d}{D})^{1/3}$ when the constraint graph is D-regular.

More recent and closely related work is that of Gharibian and Parekh [14] and Bravyi, Gosset, König, and Temme [8]. These works and our work all aim to optimize 2-local Hamiltonians in the product state space, and they all use SDP to do so. The work [14] considers a specialization of Max-2-LH; however, a key difference is that their main target Hamiltonian is more restricted and physically motivated, a maximization version of the quantum Heisenberg model. The special case they consider is qualitatively simpler in some sense, as their Hamiltonian does not have any 1-local terms, which present an additional complication in our work. They also consider additional Hamiltonians that do not have PSD local terms, which are not special cases of MAX-2-LH. Another difference is in the SDP relaxations employed. We must use additional constraints yielding a strengthened SDP relaxation to obtain our results. They get the following results.

▶ **Theorem 6** ([14]). *Consider a local Hamiltonian $H = \sum_{(p,q)\in E} w_{pq} H_{pq}$ for $w_{pq} \geq 0$, where $H_{pq} = I - \alpha X_p X_q - \beta Y_p Y_q - \gamma Z_p Z_q$, for $\alpha, \beta, \gamma \in \{0, 1\}$. There exists a randomized classical approximation algorithm with approximation ratio 0.498 (when $\alpha + \beta + \gamma = 3$), 0.649 (when $\alpha + \beta + \gamma = 2$), or 0.878 (when $\alpha + \beta + \gamma = 1$).*

The $\alpha + \beta + \gamma = 3$ case of the above problem is when H_{pq} is a rank 1 projector onto the Bell state $|\Psi^-\rangle = |01\rangle - |10\rangle$, on qubits p and q. Thus this case is an instance of MAX-2-LH.

The work [8] uses similar techniques for a slightly different problem. They consider the problem of approximating the maximum eigenvalue of traceless 2-local Hamiltonians on n qubits. They give a randomized classical approximation algorithm that outputs a product state with expected energy $\mathrm{OPT}/O(\log n)$ where OPT is the maximum eigenvalue of the input Hamiltonian. They use the same SDP relaxation as [14] for the maximum eigenvalue, but they use a different rounding technique from [14] or this work; as with [14], being able to ignore 1-local terms simplifies their algorithm and analysis relative to this work.

Another related work is by Bansal, Bravyi, and Terhal, where they prove that a PTAS (an algorithm that runs in polynomial time in problem size and $1/\epsilon$ where ϵ is an arbitrarily small approximation ratio) exists [5] for Quantum Ising Spin Glass defined over a planar interaction graph with bounded degree, where the goal is to find the minimum eigenvalue of Hamiltonians of form $H = \sum_{(u,v)\in E} c_{uv} L_{uv} + \sum_u L_u$, where L_{uv} is quadratic in Pauli matrices and L_u is linear.

There are limited results on the hardness of approximation of quantum problems. Gharibian and Kempe introduce a quantum version of Σ_2^p called cq-Σ_2 [13]. They prove that QSSC and QIRR, defined in the paper, are cq-Σ_2-hard to approximate to certain ratios. Moreover, they also show that it is QCMA-hard to approximate Quantum Monotone Minimum Satisfying Assignment, defined in their paper, to approximation ratio $N^{1-\epsilon}$ for all $\epsilon > 0$, where N is the size of instance.

2.2 Reducing Max-2-CSP to Max-2-LH

MAX-2-CSP is reduced to MAX-2-Local Hamiltonian as follows. Suppose we are given a MAX-2-CSP instance on n boolean variables $x_1, x_2, \ldots, x_n \in \{0, 1\}$, a set of edges E between the x_i's, and functions $f_{ij} : \{0, 1\}^2 \to \{0, 1\}$ on (x_i, x_j) for $(i, j) \in E$. The question is to compute the quantity $\mathrm{OPT}_{\mathrm{CSP}} = \max_{x_1,\ldots,x_n} \sum_{(i,j)\in E} f_{ij}(x_i, x_j)$. We can reduce this instance to a MAX-2-LH instance on n qubits $q_1, q_2, \ldots, q_n$ with projectors $P_{ij} := \sum_{(x_i,x_j)\in \mathrm{Supp}(f_{ij})} |x_i x_j\rangle\langle x_i x_j|$ on qubit i, j for $(i, j) \in E$, where $\mathrm{Supp}(f_{ij}) = \{(x_i, x_j) \mid f_{ij}(x_i, x_j) = 1\}$. To see that this is a correct reduction, consider an optimizer $|\phi\rangle = \sum_{x\in\{0,1\}^n} c_x |x\rangle$ to the MAX-2-LH instance. The energy of $|\phi\rangle$ is $\langle\phi| \sum_{(i,j)\in E} P_{ij} |\phi\rangle = \sum_{x\in\{0,1\}^n} |c_x|^2 \langle x| \sum_{(i,j)\in E} P_{ij} |x\rangle$. Measuring $|\phi\rangle$ in the computational basis and using it as

an assignment for the original instance, we satisfy $\sum_{x\in\{0,1\}^n} |c_x|^2 \sum_{(i,j)\in E} f_{ij}(x_i, x_j)$ constraints in expectation, which coincides with the energy of $|\phi\rangle$. So MAX-2-Local Hamiltonian on qubits is NP-hard. Then MAX-k-Local Hamiltonian with $k > 2$ on qudits is also NP-hard because MAX-2-LH on qubits is a special case of MAX-k-LH on qudits, when each projector acts non-trivially on 2 locations within an embedded 2 dimensional space.

2.3 Approximating 2-CSPs using SDP

SDP has been a major tool for approximating 2-CSP problems since Goemans and Williamson used semidefinite programming to obtain a 0.878-approximation for MAX-CUT and MAX-2-SAT, as well as a 0.796-approximation for a MAX-DICUT [15].

There had been gradual improvements in the approximation ratio [23, 22], and finally [20] obtained the best approximation ratios to date, 0.94016 for MAX-2-SAT and 0.87401 for MAX-DICUT. All of the papers mentioned above employ SDP rounding techniques.

On the other hand, [17] showed that the approximation ratio 0.878 for MAX-CUT is tight and the approximation ratio for MAX-DICUT is upper bounded by 0.878, assuming that the Unique Game Conjecture (UGC) is true. Austrin proved that the approximation ratio 0.94016 is tight for MAX-2-SAT and the approximation ratio for MAX-2-AND is upper bounded by 0.87435, assuming the UGC [4]. More details about CSP approximation can be found in a recent survey by Makarychev and Makarychev [21].

2.4 Lemmas on randomized rounding

We use the following lemmas without proof. The inequalities are special cases of more general "Grothendieck inequalities." Grothendieck inequalities have begun playing an important role in theoretical computer science with a variety of applications. For more details see a survey by Khot and Naor [18].

The first two lemmas below are given by Goemans and Williamson with which they prove approximation ratios for their MAX-CUT and MAX-2SAT algorithms respectively, namely, Theorem 3.3 and Lemma 7.3.2 in [15].

▶ **Lemma 7** ([15]). *Let $u, v \in S^N$ be unit vectors in $\mathbb{R}^{N+1}$ for $N \geq 2$, and let $x = \operatorname{sgn}(u \cdot r), y = \operatorname{sgn}(v \cdot r)$ for a uniformly random vector $r \in S^N$. Then*

$$\mathbb{E}_r[1 \pm xy] \geq \alpha_1(1 \pm u \cdot v),$$

where $\alpha_1 = \frac{2}{\pi} \min_{0<\theta<\pi} \frac{\theta}{1-\cos\theta} = 0.878\ldots$.

▶ **Lemma 8** ([15]). *Let $u, v, w \in S^N$ be unit vectors in $\mathbb{R}^{N+1}$ for $N \geq 2$, and let $x = \operatorname{sgn}(u \cdot r), y = \operatorname{sgn}(v \cdot r), z = \operatorname{sgn}(w \cdot r)$, for a uniformly random vector $r \in S^N$. Then*

$$\mathbb{E}_r[1 \pm xy \pm xz + yz] \geq \alpha_2(1 \pm u \cdot v \pm u \cdot w + v \cdot w),$$

where $\alpha_2 = \min_{0<\theta<\arccos(-1/3)} \frac{2}{\pi} \frac{2\pi-3\theta}{1+3\cos\theta} = 0.796\ldots$.

We will also use the following lemma, obtained by Briët, de Oliveira Filho, and Vallentin [9] in considering a rank-constrained version of the Grothendieck problem, to bound randomized rounding for positive semidefinite matrices.

▶ **Lemma 9** ([9]). *Let A be a $m \times m$ real-valued positive semidefinite matrix and $u_1, \ldots, u_m$ be unit vectors in S^N for an integer $N \geq m$. For all $1 \leq i \leq m$, let $x_i = \operatorname{sgn}(r \cdot u_i)$ for a uniformly random vector $r \in S^N$. Then*

$$\mathbb{E}\left[\sum_{1\leq i,j\leq m} A_{ij}x_ix_j\right] \geq \frac{m}{\pi}\left(\frac{\Gamma(m/2)}{\Gamma((m+1)/2)}\right)^2 \sum_{1\leq i,j\leq m} A_{ij}u_i \cdot u_j.$$

The coefficient $\frac{m}{\pi}\left(\frac{\Gamma(m/2)}{\Gamma((m+1)/2)}\right)^2$ is asymptotically $2/\pi + \Theta(1/m)$. We will use the lemma for $m = 6$, which has associated constant $\alpha_7 := \frac{6}{\pi}\left(\frac{\Gamma(3)}{\Gamma(3.5)}\right)^2 = \frac{6}{\pi}\left(\frac{2}{15\sqrt{\pi}/8}\right)^2 = 0.691\cdots$.

Since Lemma 9 is not explicitly stated in the form of a theorem in [9], we give a brief description about how to piece the statements in [9] together. This lemma follows from the analysis in [9], where they give an approximation algorithm for the Grothendieck problem. In particular, Lemma 1 and the argument surrounding Equation (3) are used as follows. Using their notation, let $E_1(t) = \frac{2}{\pi}\arcsin t$ and let $\widetilde{E}_1(t) = \frac{2}{\pi}\arcsin t - \frac{2}{\pi}\frac{t}{\gamma(m)}$. It turns out that the functions only depend on the inner product between two vectors, so $E_1(u_i, u_j)$ means $E_1(t)$, where $t = u_i \cdot u_j$. Lemma 1 states that $\widetilde{E}_1(t)$ is of positive type for S^{m-1}. This means that for any vectors $u_1, \ldots, u_m \in S^{m-1}$, the matrix $\left(\widetilde{E}_1(u_i, u_j)\right)_{1\leq i,j\leq m}$ is positive semidefinite. Since A is also positive semidefinite, it holds that $\sum_{i,j=1}^m A_{ij}\left(\frac{2}{\pi}\arcsin u_i \cdot u_j - \frac{2}{\pi}\frac{u_i\cdot u_j}{\gamma(m)}\right) = \sum_{i,j=1}^m A_{ij}\widetilde{E}_1(u_i, u_j) \geq 0$. Therefore $\sum_{ij} A_{ij}\frac{2}{\pi}\arcsin u_i \cdot u_j \geq \sum_{ij} A_{ij}\frac{2}{\pi}\frac{u_i\cdot u_j}{\gamma(m)}$. Finally, use the fact that $\mathbb{E}\left[\operatorname{sgn}(r \cdot u_i)\operatorname{sgn}(r \cdot u_j)\right] = \frac{2}{\pi}\arcsin u_i \cdot u_j$, and the fact that $\frac{2}{\pi}\frac{1}{\gamma(m)}$ is equal to the factor given in the lemma above, as in Theorem 2 in [9].

2.5 Notation

▶ **Definition 10** (Pauli matrices). *We denote the Pauli matrices on a qubit as $W_0 := I = \begin{pmatrix}1 & 0\\ 0 & 1\end{pmatrix}$, $W_1 := X = \begin{pmatrix}0 & 1\\ 1 & 0\end{pmatrix}$, $W_2 := Y = \begin{pmatrix}0 & -i\\ i & 0\end{pmatrix}$, and $W_3 := Z = \begin{pmatrix}1 & 0\\ 0 & -1\end{pmatrix}$. On n qubits, we write $W_{pi} := W_i \otimes I_{[n]\setminus\{p\}}$, where W_i is on the qubit p, and $I_{[n]\setminus\{p\}}$ is the identity on the rest of the qubits.*

3 The algorithm and analysis

The setup

A 2-local Hamiltonian is given as a set of Hamiltonians $\{H_{pq} : (p, q) \in E\}$, for some edge set $E \subseteq [n] \times [n]$, where H_{pq} operates on qubits p and q. This means H_{pq} is a 4×4 matrix on the Hilbert space $\mathcal{H}_p \otimes \mathcal{H}_q$ that represents the joint space of qubits p, q. Given such a list of Hamiltonians on 2 qubits, the goal is to approximate the maximum eigenvalue of $H := \sum_{(p,q)\in E} H_{pq} \otimes I_{[n]\setminus\{p,q\}}$. In this paper we assume each term H_{pq} is a PSD matrix. We will decompose H_{pq} using its eigenvectors $H_{pq} = \sum_{t=1}^4 w_{pq,t}|\gamma_{pq,t}\rangle\langle\gamma_{pq,t}|$, where $w_{pq,i} \geq 0$ for all p, q, t.

Let OPT denote the maximum eigenvalue of H over quantum states. Stated in terms of energy, $\mathrm{OPT} = \max_{|\phi\rangle}\langle\phi|H|\phi\rangle$. We seek to produce an approximately optimal state; however, since our approximation algorithms are classical polynomial-time algorithms, we output an efficient representation of such a state. Our approach is to formulate an SDP that upper-bounds the maximum energy of the given Hamiltonian. We then randomly round the solution vectors of the SDP to real numbers to get a product quantum state, admitting a polynomial-size representation, with a relatively high energy.

A semidefinite program relaxation

We start with an SDP relaxation that gives an upper bound on the optimal energy. By "relaxation", we mean an SDP for which each quantum state generates a feasible solution, with objective function value matching its energy. This way, we know that the value of the SDP is at least the highest energy possible by a physical state.

For each pair of qubits $(p,q) \in E$ (E consists of ordered pairs pq with $p<q$), we have a local Hamiltonian $H_{pq} = \sum_{t=1}^{4} w_{pq,t}|\gamma_{pq,t}\rangle\langle\gamma_{pq,t}|$. We denote the Pauli decomposition of the local terms $C_{pqij} = \mathrm{Tr}(H_{pq}(W_i \otimes W_j))/4$. Note that the matrix $C_{pq} = (C_{pqij})_{ij}$ is real for all $(p,q) \in E$, since tensor products of Pauli matrices form a real-coefficient basis for the Hermitian matrices. Consider the SDP:

$$\text{Maximize} \quad \sum_{(p,q)\in E} \sum_{i,j=0}^{3} C_{pqij} v_{pi} \cdot v_{qj} \tag{S1}$$

subject to:

$$\begin{aligned}
&||v_0|| = 1 \\
&||v_{pi}|| = 1, \ \forall p, i \\
&v_0 \cdot v_{p0} = 1, \ \forall p \\
&v_{pi} \cdot v_{pj} = 0, \ \forall p, \ i,j > 0 : i \neq j \\
&v_0, v_{pi} \in \mathbb{R}^N, \ \forall p, i \\
&\sum_{i,j=0}^{3} C_{pqij} v_{pi} \cdot v_{qj} \leq \sum_{t=1}^{4} w_{pq,t}, \ \forall (p,q) \in E.
\end{aligned}$$

and N can be taken to be the total number of vectors, v_0 and the v_{pi} in the program, which is $4n+1$. The motivation for this relaxation is that, for any quantum state $|\phi\rangle$, we may construct additional complex unit vectors by applying tensor products of Pauli matrices to $|\phi\rangle$. In particular, setting $v_0 := |\phi\rangle$, and $v_{pi} := W_{pi}|\phi\rangle$, for all p and i, yields a solution that is "almost" feasible for the above SDP. The first three constraints are satisfied, noting that $W_{p0} = W_0 = I$ for all p. For the fourth constraint, we have that $v_{pi} \cdot v_{pj} = \langle\phi|W_{pi}W_{pj}|\phi\rangle = -\langle\phi|W_{pj}W_{pi}|\phi\rangle$, since distinct nontrivial Pauli matrices anticommute. This implies that $\mathrm{Re}(v_{pi} \cdot v_{pj}) = 0$. Indeed, we obtain a true solution to the SDP by modifying the vectors so that the imaginary part of their inner products is discarded. This is formalized in the claim below.

▶ Remark 11. The complex vectors v_{pi}, as constructed above from $|\phi\rangle$, suggest a complex SDP relaxation; however, the real version, ($S1$) is simpler and equivalent in terms of optimal objective value. The relationship $v_{pi} \cdot v_{qj} = \langle\phi|W_{pi}W_{qj}|\phi\rangle$ connects 2-moments of the state $|\phi\rangle$ to SDP relaxations. Considering higher-order moments leads to larger but stronger SDP relaxations, and considering n-moments leads to an exact SDP formulation of exponential size. Restricting the moments considered to products over only $W_{p3} = Z_p$ for different p yields an SDP hierarchy equivalent to the celebrated classical Lasserre hierarchy.

▷ Claim 12. This SDP is a relaxation, i.e., there is a mapping from pure quantum states to vectors such that objective value is the energy of the state.

Proof. Let $|\phi\rangle$ be a quantum state, and let $v_0 := (\mathrm{Re}(|\phi\rangle), \mathrm{Im}(|\phi\rangle))$, and $v_{pi} := (\mathrm{Re}(W_{pi}|\phi\rangle), \mathrm{Im}(W_{pi}|\phi\rangle))$, for $p \in [n], i \in \{0,1,2,3\}$. Here $\mathrm{Re}(u) := (\mathrm{Re}(u_i))_{i\in[d]}$, $\mathrm{Im}(u) := (\mathrm{Im}(u_i))_{i\in[d]}$ for a d-dimensional complex vector u.

To check the SDP constraints, first note the v_0 has norm 1, and $v_0 = v_{p0}$ for all p. Next, each vector v_{pi} has norm 1, and two different vectors on qubit p are orthogonal:

$$\begin{aligned} v_{pi} \cdot v_{pj} &= (\mathrm{Re}(W_{pi}|\phi\rangle), \mathrm{Im}(W_{pi}|\phi\rangle)) \cdot (\mathrm{Re}(W_{pj}|\phi\rangle), \mathrm{Im}(W_{pj}|\phi\rangle)) \\ &= \mathrm{Re}(\langle\phi|W_{pi}W_{pj}|\phi\rangle) = \delta_{ij}, \end{aligned}$$

for all $i, j \in \{1, 2, 3\}$, $p \in [n]$. This follows from the Pauli matrices anticommutation relations $W_iW_j + W_jW_i = 2\delta_{ij}I$.

Next, we consider two vectors on different qubits. Let $\rho := |\phi\rangle\langle\phi|$. Then

$$\begin{aligned} v_{pi} \cdot v_{qj} &= (\mathrm{Re}(W_{pi}|\phi\rangle), \mathrm{Im}(W_{pi}|\phi\rangle)) \cdot (\mathrm{Re}(W_{qj}|\phi\rangle), \mathrm{Im}(W_{qj}|\phi\rangle)) \\ &= \mathrm{Re}(\langle\phi|W_{pi}W_{qj}|\phi\rangle) = \mathrm{Re}(\mathrm{Tr}(W_{pi}W_{qj}\rho)), \end{aligned}$$

for all $p \neq q$, and $i, j \in \{0, 1, 2, 3\}$. This follows from the fact that two Pauli matrices commute when they act on different qubits.

The SDP objective is therefore equal to the enegy of the state:

$$\begin{aligned} \sum_{(p,q)\in E} \sum_{i,j=0}^{3} C_{pqij} v_{pi} \cdot v_{qj} &= \sum_{(p,q)\in E} \sum_{i,j=0}^{3} \frac{1}{4} \mathrm{Tr}(H_{pq}(W_i \otimes W_j)) \mathrm{Re}(\mathrm{Tr}(W_{pi}W_{qj}\rho)) \\ &= \mathrm{Re}\left(\sum_{(p,q)\in E} \sum_{i,j=0}^{3} \frac{1}{4} \mathrm{Tr}(H_{pq}(W_i \otimes W_j)) \mathrm{Tr}((W_i \otimes W_j)\rho_{pq}) \right) \\ &= \sum_{(p,q)\in E} \mathrm{Tr}(H_{pq}\rho_{pq}) = \mathrm{Tr}(H\rho), \end{aligned}$$

where ρ_{pq} is the reduced density matrix of ρ onto qubits $\{p, q\}$. The second last equality is because the Pauli matrices form an orthogonal basis. The last constraint of $(S1)$ asserts that the energy contribution from the site (p, q) should not exceed the sum of the eigenvalues of (p, q). The seemingly unnecessary last constraint will be useful in the analysis. ◁

We can solve the SDP to arbitrary multiplicative quality $1 - \epsilon$ using existing SDP solvers (for example, [1]), in time polynomial to the size of the program and $\log(1/\epsilon)$. This gives us a solution of quality $\mathrm{OPT}_{\mathrm{SDP}}(1 - \epsilon)$, where $\mathrm{OPT}_{\mathrm{SDP}}$ is the optimal value of the SDP. We will ignore $(1 - \epsilon)$ factor because it can be absorbed into our approximation ratio.

The algorithm and analysis

Algorithm 1 MAX-2-Local Hamiltonian with PSD local terms.

1: Input: $\{H_{pq} : (p, q) \in E\}, E \subseteq [n] \times [n]$.
2: Calculate $(C_{pqij} := \mathrm{Tr}(H_{pq}(W_i \otimes W_j))/4)_{pqij}$.
3: Solve the SDP $(S1)$, and get $(v_0, (v_{pi})_{pi})$.
4: Pick a uniformly random unit vector $r \in \mathbb{R}^N$ (use $N := 4n + 1$).
5: Set $(x_0, (x_{pi})_{pi}) := (\mathrm{sgn}(v_0 \cdot r), (\mathrm{sgn}(v_{pi} \cdot r))_{pi})$.
6: If $\sum_{(p,q)\in E} \left(\sum_{i=1}^{3} C_{pqi0} x_{pi} x_0 + \sum_{j=1}^{3} C_{pq0j} x_0 x_{qj} \right) < 0$, set $\widetilde{x}_{pi} := -x_{pi}x_0$ for all p and $1 \leq i \leq 3$. Otherwise set $\widetilde{x}_{pi} := x_{pi}x_0$ for all p, $1 \leq i \leq 3$.
7: Output state $\rho := \bigotimes_{p=1}^{n} \rho_p$, where $\rho_p := \frac{1}{2}I + \frac{1}{2} \sum_{i=1}^{3} \frac{1}{\sqrt{3}} \widetilde{x}_{pi} W_i$

Algorithm 1 describes our rounding procedure. Let $C_{pq} \in \mathbb{R}^{4\times 4}$ refer to the matrix $(C_{pqij})_{ij}$, let $c_{pq} := C_{pq00} = \frac{1}{4}\sum_{t=1}^{4} w_{pq,t}$, let $a_{pq} \in \mathbb{R}^3$ refer to the vector $(C_{pqi0})_{i:i>0}$, and $b_{pq} := (C_{pq0j})_{j:j>0}$ is defined analogously. Finally $\bar{C}_{pq} \in \mathbb{R}^{3\times 3}$ is $(C_{pqij})_{ij:i,j>0}$. So we have

$$C_{pq} = \begin{bmatrix} c_{pq} & b_{pq}^T \\ a_{pq} & \bar{C}_{pq} \end{bmatrix}.$$

To apply the bound from [9], we need a symmetrized and PSD version of C_{pq}, so define

$$C_{pq}^{+} = \begin{bmatrix} c_{pq} & a_{pq}^T & b_{pq}^T \\ a_{pq} & c_{pq} I & \bar{C}_{pq} \\ b_{pq} & \bar{C}_{pq}^T & c_{pq} I W_{pqi} \end{bmatrix} \in \mathbb{R}^{7\times 7}.$$

The matrix C_{pq}^{+} is clearly symmetic. To see that $C_{pq}^{+} \succeq 0$, use the fact that $H_{pq} = \sum_{t=1}^{4} w_{pq,t}|\gamma_{pq,t}\rangle\langle\gamma_{pq,t}|$ for non-negative $w_{pq,t}$'s, and get

$$\begin{aligned} 4W_{pqj}C_{pqij}^{+} &= \sum_{t=1}^{4} w_{pq,t}\mathrm{Re}\left(\mathrm{Tr}\left[|\gamma_{pq,t}\rangle\langle\gamma_{pq,t}|W_{pqi}W_{pqj}\right]\right) \\ &= \sum_{t=1}^{4} w_{pq,t}(\mathrm{Re}(W_{pqi}|\gamma_{pq,t}\rangle), \mathrm{Im}(W_{pqi}|\gamma_{pq,t}\rangle)) \cdot (\mathrm{Re}(W_{pqj}|\gamma_{pq,t}\rangle), \mathrm{Im}(W_{pqj}|\gamma_{pq,t}\rangle)), \end{aligned}$$

where $(W_{pqi})_{0\le i\le 6} := (I, W_1\otimes I, W_2\otimes I, W_3\otimes I, I\otimes W_1, I\otimes W_2, I\otimes W_3)$. Since C_{pq}^{+} is therefore a non-negative-weighted sum of Gram matrices, it is PSD. Now we state the main theorem.

▶ **Theorem 13.** *Algorithm 1 runs in polynomial time and outputs a product state with expected energy at least* $0.328\,\mathrm{OPT}$*, where* OPT *is the maximum energy of the 2-local Hamiltonian* $H = \sum_{(p,q)\in E} H_{pq}\otimes I_{[n]\setminus\{p,q\}}$ *where* H_{pq} *is PSD for all* $(p,q)\in E$*.*

Proof. Algorithm 1 outputs a product state, whose reduced part on p and q is

$$\rho_p\otimes\rho_q = \frac{1}{4}\left(I + \frac{\widetilde{x}_{p1}}{\sqrt{3}}W_1 + \frac{\widetilde{x}_{p2}}{\sqrt{3}}W_2 + \frac{\widetilde{x}_{p3}}{\sqrt{3}}W_3\right)\otimes\left(I + \frac{\widetilde{x}_{q1}}{\sqrt{3}}W_1 + \frac{\widetilde{x}_{q2}}{\sqrt{3}}W_2 + \frac{\widetilde{x}_{q3}}{\sqrt{3}}W_3\right).$$

Thus the total energy of the approximate solution is $\mathbb{E}_r\left[\sum_{(p,q)\in E}\mathrm{Tr}(H_{pq}(\rho_p\otimes\rho_q))\right]$, which is equal to

$$\mathbb{E}_r \sum_{(p,q)\in E}\left[c_{pq} + \frac{\sum_{i=1}^{3} a_{pqi}\widetilde{x}_{pi} + \sum_{j=1}^{3} b_{pqj}\widetilde{x}_{qj}}{\sqrt{3}} + \frac{\sum_{i,j=1}^{3}\bar{C}_{pqij}\widetilde{x}_{pi}\widetilde{x}_{qj}}{3}\right]. \tag{1}$$

Meanwhile, the SDP objective value is

$$\sum_{(p,q)\in E}\left[c_{pq} + \sum_{i=1}^{3} a_{pqi}v_{pi}\cdot v_0 + \sum_{j=1}^{3} b_{pqj}v_0\cdot v_{qj} + \sum_{i,j=0}^{3}\bar{C}_{pqij}v_{pi}\cdot v_{qj}\right]. \tag{2}$$

Fix an edge $(p,q) \in E$, and let $v_{pq} := (v_0, v_{p1}, v_{p2}, v_{p3}, v_{q1}, v_{q2}, v_{q3})$. By applying Lemma 9 to C^+_{pq} and v_{pq}, we get

$$\mathbb{E}_r \left[7c_{pq} + 2\sum_{i=1}^{3} a_{pqi} x_{pi} x_0 + 2\sum_{j=1}^{3} b_{pqj} x_0 x_{qj} + 2\sum_{i,j=1}^{3} \bar{C}_{pqij} x_{pi} x_{qj} \right] \geq$$
$$\alpha_7 \left(7c_{pq} + 2\sum_{i=1}^{3} a_{pqi} v_{pi} \cdot v_0 + 2\sum_{j=1}^{3} b_{pqj} v_0 \cdot v_{qj} + 2\sum_{i,j=1}^{3} \bar{C}_{pqij} v_{pi} \cdot v_{qj} \right). \tag{3}$$

The coefficients 7 and 2 above come from how many times each parts are in C^+_{pq}. We would like to compare the energy of our solution (1) against the SDP objective (2) using the inequality (3). For the series of inequalities that will appear below, we define some short-hand:

$$L_x := \sum_{(p,q)\in E} \left[\sum_{i=1}^{3} a_{pqi} x_{pi} x_0 + \sum_{j=1}^{3} b_{pqj} x_0 x_{qj} \right],$$
$$Q_x := \sum_{(p,q)\in E} \left[\sum_{i,j=1}^{3} \bar{C}_{pqij} x_{pi} x_{qj} \right],$$
$$L_v := \sum_{(p,q)\in E} \left[\sum_{i=1}^{3} a_{pqi} v_{pi} \cdot v_0 + \sum_{j=1}^{3} b_{pqj} v_0 \cdot v_{qj} \right],$$
$$Q_v := \sum_{(p,q)\in E} \left[\sum_{i,j=1}^{3} \bar{C}_{pqij} v_{pi} \cdot v_{qj} \right],$$
$$\widetilde{L}_x := \sum_{(p,q)\in E} \left[\sum_{i=1}^{3} a_{pqi} \widetilde{x}_{pi} + \sum_{j=1}^{3} b_{pqj} \widetilde{x}_{qj} \right],$$
$$\widetilde{Q}_x := \sum_{(p,q)\in E} \left[\sum_{i,j=1}^{3} \bar{C}_{pqij} \widetilde{x}_{pi} \widetilde{x}_{qj} \right],$$
$$c := \sum_{(p,q)\in E} c_{pq}.$$

After summing the inequality (3) over the edges and moving the terms, we get

$$\mathbb{E}_r \left[c + \frac{L_x + Q_x}{3} \right] \geq \frac{\alpha_7(\frac{7}{2}c + L_v + Q_v) - \frac{1}{2}c}{3}. \tag{4}$$

Because $\widetilde{L}_x \geq 0$ (Step 6 of Algorithm 1), and $\widetilde{L}_x + \widetilde{Q}_x \geq L_x + Q_x$ for every r, we can bound the energy of the approximate solution:

$$\mathbb{E}_r \left[c + \frac{\widetilde{L}_x}{\sqrt{3}} + \frac{\widetilde{Q}_x}{3} \right] \geq \mathbb{E}_r \left[c + \frac{\widetilde{L}_x + \widetilde{Q}_x}{3} \right] \geq \mathbb{E}_r \left[c + \frac{L_x + Q_x}{3} \right] \geq \frac{\alpha_7(\frac{7}{2}c + L_v + Q_v) - \frac{1}{2}c}{3},$$

where the last inequality follows from (4). We can now bound the approximation factor,

$$\frac{\mathbb{E}_r \left[c + \frac{\widetilde{L}_x}{\sqrt{3}} + \frac{\widetilde{Q}_x}{3} \right]}{c + L_v + Q_v} \geq \frac{\alpha_7(\frac{7}{2}c + L_v + Q_v) - \frac{1}{2}c}{3(c + L_v + Q_v)} = \frac{\alpha_7(\frac{7}{2}c + S) - \frac{1}{2}c}{3(c + S)},$$

letting $S := L_v + Q_v$ for convenience. The value of the SDP $(S1)$ is $c + S$ and the energy of the maximally mixed state is $c = \sum_{(p,q)\in E} c_{pq}$. The SDP value $c + S$ should at least be the energy of the maximally mixed state, so $c + S \geq c$. This implies that $S \geq 0$. By summing the last constraint of $(S1)$ over the edges, we get $c + S \leq \sum_{(p,q)\in E, 1\leq t\leq 4} w_{pq,t} = 4c$. Therefore, the approximation ratio is at least

$$\min_{S\in[0,3c]} \frac{\alpha_7(\frac{7}{2}c + S) - \frac{1}{2}c}{3(c + S)} \geq \frac{\alpha_7(\frac{13}{8}) - \frac{1}{8}}{3} = 0.328\ldots.$$

The min occurs at $S = 3c$ since $\frac{\alpha_7(\frac{7}{2}c+S)-\frac{1}{2}c}{3(c+S)} = \frac{\alpha_7(c+S)+\alpha_7\frac{5}{2}c-\frac{1}{2}c}{3(c+S)} = \frac{\alpha_7}{3} + \frac{5\alpha_7-1}{6(c+S)}c$ is decreasing for $c + S \geq 0$. ◀

4 Improved approximation in for tensor product projectors using a different SDP and rounding

In this section we show that a better approximation factor is possible when each local term is a rank 1 projector that is a product of 2 rank 1 projectors on a qubit. Namely, $H_{pq} = H_{pq,p} \otimes H_{pq,q}$, and $H_{pq,p}, H_{pq,q}$ are rank 1 projectors on qubit p, q respectively for all $(p, q) \in E$. The algorithm is similar to the previous one in that Goemans-Williamson approach is used. This time, however, we formulate the SDP and round the vector solution in a different way: we first formulate a quadratic programming (QP) in the real space that has the highest energy by a product state as the objective value, and relax the QP to an SDP. The previous rounding reduced the norms of the solution vectors by $1/\sqrt{3}$, but here we will keep the norms. The first observation we make is that the value of the following quadratic program (Q) is the maximum energy achieved by a product state.

Quadratic program for MAX-2-Local Hamiltonian over product states:

$$\text{Maximize } 4\sum_{pq}\sum_{i,j=0}^{3} C_{pqij}x_{pi}x_{qj} \tag{Q}$$

subject to:

$$x_{p0} = \frac{1}{2},\ \forall p$$

$$\sum_{i=0}^{3} x_{pi}^2 = \frac{1}{2},\ \forall p$$

$$x_{pi} \in \mathbb{R},\ \forall p, i.$$

The Pauli coefficients C_{pqij} are the same as defined previously. The constraints will restrict us to the set of quantum states that are product states. We use the Pauli basis so that all numbers we solve for are real valued. Representing qubits as density matrices, let $\Phi_p := |\phi_p\rangle\langle\phi_p|$ for all p.

For any density matrix, there exists a unique decomposition into Pauli matrices. So we can write $\Phi_p = \sum_{i=0}^{3} x_{pi}W_i$, and $H_{pq} = \sum_{i,j=0}^{3} C_{pqij}W_i \otimes W_j$. Moreover, we know that $\mathrm{Tr}[\Phi_p] = 1$ because the state $|\phi_p\rangle$ is norm 1, and also $\mathrm{Tr}[\Phi_p^2] = 1$ if we assume, without loss of generality, that the state $|\phi_p\rangle$ is pure. This implies that (i) $x_{p0} = \frac{1}{2}$ for all p, (ii) $\sum_i x_{pi}^2 = \frac{1}{2}$ for all p, (iii) $C_{pq00} = \frac{1}{4}$ for all p, q, and (iv) $\sum_{ij} C_{pqij}^2 = \frac{1}{4}$ for all p, q.

The reverse process also works: Given real numbers $(x_{p0}, x_{p1}, x_{p2}, x_{p3})$ as a part of a feasible solution to (Q), one can construct a state $|\phi_p\rangle$ such that $|\phi_p\rangle\langle\phi_p| = \sum_{i=0}^{3} x_{pi}W_i$.

This is because of the properties of the Pauli basis:

$$\begin{aligned}\Phi_p^2 &= \sum_{i,j=0}^{3} x_{pi}x_{pj}W_iW_j = x_{p0}^2W_0^2 + 2\sum_{i=0}^{3} x_{pi}x_{p0}W_i + \sum_{i,j=1}^{3} x_{pi}x_{pj}W_iW_j \\ &= \frac{1}{4}W_0 + \sum_{i=0}^{3} x_{pi}W_i + \sum_{i=1}^{3} x_{pi}^2W_i^2 + \sum_{1\le i,j\le 3, i\neq j}^{3} x_{pi}x_{pj}W_iW_j \\ &= \frac{1}{4}W_0 + \sum_{i=0}^{3} x_{pi}W_i + \sum_{i=1}^{3} x_{pi}^2W_0 + \sum_{1\le i<j\le 3}^{3} x_{pi}x_{pj}(W_iW_j + W_jW_i) \\ &= \frac{1}{4}W_0 + \sum_{i=0}^{3} x_{pi}W_i + \frac{1}{4}W_0 + 0 = \Phi_p.\end{aligned}$$

The fact that $\Phi_p^2 = \Phi_p$ implies that Φ_p is a projector, and $\operatorname{Tr}\Phi_p = 1$ implies that it is a rank 1 projector. So there exists a vector $|\phi_p\rangle$ such that $|\phi_p\rangle\langle\phi_p| = \Phi_p = \sum_{i=0}^{3} x_{pi}W_i$.

Exactly solving this program is NP-hard, because as noted in Section 2.2, MAX-2-CSP can be cast as an instance of MAX-2-LH. Additionally, an optimum solution to such a MAX-2-LH instance is always, without loss of generality, a product state. We can, however, solve this approximately by first relaxing the program to an SDP, solving the SDP in polynomial time, and performing randomized rounding to the SDP solution we get to obtain a valid solution to the original program. Below is the SDP to which we relax. The coefficients C_{pqij}'s are given as constants.

SDP relaxation for MAX-2-Local Hamiltonian over product states:

$$\text{Maximize } 4\sum_{pq}\sum_{i,j=0}^{3} C_{pqij}v_{pi}\cdot v_{qj} \tag{S2}$$

subject to:

$$\begin{aligned}&\|v_0\| = \frac{1}{2} \\ &\|v_{p0}\| = \frac{1}{2},\ \forall p \\ &v_0\cdot v_{p0} = \frac{1}{4},\ \forall p \\ &\sum_{i=0}^{3}\|v_{pi}\|^2 = \frac{1}{2},\ \forall p \\ &v_0 \in \mathbb{R}^N \\ &v_{pi} \in \mathbb{R}^N,\ \forall p, i.\end{aligned}$$

The first three conditions simply force that for all p, $v_{p0} = v_0$. The SDP is in fact a relaxation of the quadratic program in the sense that given a solution x_{pi}, for all p, i, to the first program, the vector solution $v_0 := (1, 0, \ldots, 0), v_{pi} := x_{pi}v_0$, for all p, i, is a feasible solution to the SDP that achieves the same objective value. Therefore the value of the SDP is at least the value of the first program. Algorithm 2 details our approach.

▶ **Theorem 14.** *Given a local Hamiltonian on n qubits $H = \sum_{pq} P_{pq}$, where P_{pq} is a product of a rank 1 projector on qubit p and a rank 1 projector on qubit q, Algorithm 2 outputs a product state that has an expected energy of at least $\alpha_2 \cdot \mathrm{OPT}_{\mathrm{prod}}$, where $\alpha_2 = 0.796\ldots$, and $\mathrm{OPT}_{\mathrm{prod}} = \max_{|\phi_1\rangle,\ldots,|\phi_n\rangle}\langle\phi_1|\ldots\langle\phi_n|H|\phi_1\rangle\ldots|\phi_n\rangle$.*

Algorithm 2 MAX-2-local Hamiltonian with PSD local terms.

1: Input: $\{H_{pq} : (p,q) \in E\}, E \subseteq [n] \times [n]$.
2: Calculate $(C_{pqij} := \mathrm{Tr}(H_{pq}(W_i \otimes W_j))/4)_{pqij}$.
3: Solve the SDP $(S2)$, and get $(v_0, (v_{pi}))$.
4: Pick a random unit vector $r \in \mathbb{R}^N$ (use $N := 4n+1$).
5: For all $p \in [n]$ and $i \in \{0,1,2,3\}$, assign $x_{pi} := \|v_{pi}\| \operatorname{sgn}(v_0 \cdot r) \operatorname{sgn}(v_{pi} \cdot r)$.
6: Output $\Phi_p := \bigotimes_{p=1}^n \sum_{i=0}^3 x_{pi} W_i$ as the resulting product state assignment.

Proof. We express the rank 1 projector H_{pq} as $|\gamma_{pq}\rangle\langle\gamma_{pq}|$ for some state $|\gamma_{pq}\rangle$. $\mathrm{OPT}_{\mathrm{prod}}$ is the value of the program (Q). We relax the program (Q) to the SDP $(S2)$ and get v_{pi} for all p, i as a solution to the SDP. To get real-valued x_{pi}, we perform randomized rounding reminiscent of the Goemans-Williamson algorithm: $x_{pi} = \|v_{pi}\| \operatorname{sgn}(v_0 \cdot r) \operatorname{sgn}(v_{pi} \cdot r)$. The x_{pi} are used in Step 6 of the algorithm to produce a product state; these variables constitute a feasible solution to (Q) because $x_{p0} = \|v_{p0}\| \operatorname{sgn}(v_0 \cdot r) \operatorname{sgn}(v_{p0} \cdot r) = \frac{1}{2} \operatorname{sgn}(v_0 \cdot r)^2 = \frac{1}{2}$, and

$$\sum_{i=0}^3 x_{pi}^2 = \sum_{i=0}^3 \|v_{pi}\|^2 \operatorname{sgn}(v_0 \cdot r)^2 \operatorname{sgn}(v_{pi} \cdot r)^2 = \sum_{i=0}^3 \|v_{pi}\|^2 = \frac{1}{2}.$$

We apply Lemma 16 below to analyze the performance of the rounding term by term to yield:

$$\mathbb{E}\left[4 \sum_{pq} \sum_{i,j=0}^3 C_{pqij} x_{pi} x_{qj}\right] \geq \alpha_2 4 \sum_{pq} \sum_{i,j=0}^3 C_{pqij} v_{pi} \cdot v_{qj} = \alpha_2 \,\mathrm{OPT}_{\mathrm{SDP}} \geq \alpha_2 \,\mathrm{OPT}_{\mathrm{prod}}\,. \blacktriangleleft$$

▶ **Corollary 15.** *Algorithm 2 outputs a product state achieving* $0.40 \cdot \mathrm{OPT}$.

Proof. By Theorem 14 the algorithm outputs a state achieving $0.796 \cdot \mathrm{OPT}_{\mathrm{prod}}$, which is at least $0.40 \cdot \mathrm{OPT}\,/2$ by Theorem 3. ◀

Note that we have not yet used the assumption that H_{pq} is a tensor product of rank 1 projectors. In this case we have that $|\gamma_{pq}\rangle\langle\gamma_{pq}| = H_{pq} = H_{pq,p} \otimes H_{pq,q} = |\gamma_p\rangle\langle\gamma_p| \otimes |\gamma_q\rangle\langle\gamma_q| = (|\gamma_p\rangle \otimes |\gamma_q\rangle)(\langle\gamma_p| \otimes \langle\gamma_q|)$, for some states $|\gamma_p\rangle$ and $|\gamma_q\rangle$. Consequently $|\gamma_{pq}\rangle = |\gamma_p\rangle \otimes |\gamma_q\rangle$ is a product state, and the Pauli coefficients are $C_{pqij} = \mathrm{Tr}(|\gamma_{pq}\rangle\langle\gamma_{pq}|W_i \otimes W_j)/4 = \mathrm{Tr}((|\gamma_p\rangle\langle\gamma_p| \otimes |\gamma_q\rangle\langle\gamma_q|)(W_i \otimes W_j))/4 = \mathrm{Tr}(|\gamma_p\rangle\langle\gamma_p|W_i)\,\mathrm{Tr}(|\gamma_p\rangle\langle\gamma_p|W_i)/4 = C_{pi}C_{qj}$. So for all i, j, $C_{pqij} = C_{pi}C_{qj}$. Note that since $|\gamma_p\rangle$ is a 1 qubit state, $C_{p0}^2 = (\frac{1}{2})^2 = \sum_{i=1}^3 C_{pi}^2$. To conclude our analysis, we leverage this structure in the lemma below.

▶ **Lemma 16.** *Let* $u_0, \ldots, u_3, v_0, \ldots, v_3 \in \mathbb{R}^N$ *be vectors such that* $u_0 = v_0$, $\|u_0\| = \|v_0\| = 1/2$, *and* $\sum_{i=0}^3 \|u_i\|^2 = \sum_{i=0}^3 \|v_i\|^2 = 1/2$. *Let* $x_i = \|u_i\| \operatorname{sgn}(u_0 \cdot r) \operatorname{sgn}(u_i \cdot r)$, $y_j = \|v_j\| \operatorname{sgn}(v_0 \cdot r) \operatorname{sgn}(v_j \cdot r)$ *be the rounding of the vectors with respect to a uniformly random vector* $r \in S^{N-1}$. *Let* $C_{ij} = C_i D_j$ *where* $C_i, D_j \in \mathbb{R}$ *such that* $C_0^2 = \sum_{i=1}^3 C_i^2 = D_0^2 = \sum_{i=1}^3 D_i^2$. *Then*

$$\alpha_2 \sum_{i,j=0}^3 C_{ij} u_i \cdot v_j \leq \mathbb{E}\left[\sum_{i,j=0}^3 C_{ij} x_i y_j\right], \text{ where } \alpha_2 = \min_{0<\theta<\arccos -1/3} \frac{2}{\pi} \frac{2\pi - 3\theta}{1 + 3\cos\theta} = 0.796\ldots.$$

Proof. The proof is by applying Lemma 7, Lemma 8, and the Cauchy-Schwarz inequality. Let $U_i = \operatorname{sgn}(u_i \cdot r)$ and $V_i = \operatorname{sgn}(v_i \cdot r)$ and note that $U_0 = V_0$. Also for convenience, set

$$A_0 := C_0\|u_0\| - \sum_{i=1}^{3} |C_i|\|u_i\|, \quad A_1 := \sum_{i=1}^{3} |C_i|\|u_i\|(1 + \operatorname{sgn}(C_i)U_0U_i),$$
$$B_0 := D_0\|v_0\| - \sum_{j=1}^{3} |D_j|\|v_j\|, \quad B_1 := \sum_{i=1}^{3} |C_i|\|u_i\|(1 + \operatorname{sgn}(C_i)U_0U_i)$$

$$\begin{aligned}
\mathbb{E}\left[\sum_{i,j=0}^{3} C_{ij}x_iy_j\right] &= \mathbb{E}\left[\sum_{i,j=0}^{3} C_iD_j\|u_i\|U_0U_i\|v_j\|V_0V_j\right] \\
&= \mathbb{E}\left[\sum_{i=0}^{3} C_i\|u_i\|U_0U_i \sum_{j=0}^{3} D_j\|v_j\|V_0V_j\right] \\
&= \mathbb{E}\left[\left(C_0\|u_0\| + \sum_{i=1}^{3} C_i\|u_i\|U_0U_i\right)\left(D_0\|v_0\| + \sum_{j=1}^{3} D_j\|v_j\|V_0V_j\right)\right] \\
&= \mathbb{E}\left[(A_0 + A_1)(B_0 + B_1)\right] \\
&= A_0B_0 + \mathbb{E}\left[A_1B_0\right] + \mathbb{E}\left[A_0B_1\right] + \mathbb{E}\left[A_1B_1\right].
\end{aligned}$$

For convenience, set

$$A_0' := A_0, \quad A_1' := \sum_{i=1}^{3} |C_i|\|u_i\|(1 + \operatorname{sgn}(C_i)\frac{u_0 \cdot u_i}{\|u_0\|\|u_i\|}),$$
$$B_0' := B_0, \quad B_1' := \sum_{i=1}^{3} |D_i|\|v_i\|(1 + \operatorname{sgn}(D_i)\frac{u_0 \cdot v_i}{\|u_0\|\|v_i\|}).$$

From Lemma 7,

$$\mathbb{E}\left[1 + \operatorname{sgn}(C_i)\operatorname{sgn}(u_0 \cdot r)\operatorname{sgn}(u_i \cdot r)\right] \geq \alpha_1\left(1 + \operatorname{sgn}(C_i)\frac{u_0 \cdot u_i}{\|u_0\|\|u_i\|}\right)$$

for all i. Therefore $\mathbb{E}\left[A_1\right] \geq \alpha_1 A_1'$ and $\mathbb{E}\left[B_1\right] \geq \alpha_1 B_1'$.

Using the fact that either exactly one of $\operatorname{sgn}(C_i)$, $\operatorname{sgn}(D_j)$, or $\operatorname{sgn}(C_i)\operatorname{sgn}(D_j)$ is positive, or all three are, Lemma 8 implies

$$\begin{aligned}
&\mathbb{E}\left[(1 + \operatorname{sgn}(C_i)\operatorname{sgn}(u_0 \cdot r)\operatorname{sgn}(u_i \cdot r))(1 + \operatorname{sgn}(D_j)\operatorname{sgn}(v_0 \cdot r)\operatorname{sgn}(v_j \cdot r))\right] \geq \\
&\alpha_2\left(1 + \operatorname{sgn}(C_i)\frac{u_0 \cdot u_i}{\|u_0\|\|u_i\|} + \operatorname{sgn}(D_j)\frac{v_0 \cdot v_j}{\|v_0\|\|v_j\|} + \operatorname{sgn}(C_iD_j)\frac{u_i \cdot v_j}{\|u_i\|\|v_j\|}\right)
\end{aligned}$$

for all i, j. Therefore $\mathbb{E}\left[A_1B_1\right] \geq \alpha_2 A_1'B_1'$.

From Cauchy-Schwarz we have $A_0, B_0 \geq 0$, and using this inequality we can bound

$$\begin{aligned}
&\mathbb{E}\left[\sum_{i,j=0}^{3} C_{ij} x_i y_j\right] \\
&= A_0 B_0 + \mathbb{E}\left[A_1 B_0\right] + \mathbb{E}\left[A_0 B_1\right] + \mathbb{E}\left[A_1 B_1\right]. \\
&\geq A_0 B_0 + \alpha_1 A_1' B_0' + \alpha_1 A_0' B_1' + \alpha_2 A_1' B_1' \\
&\geq \alpha_2 A_0 B_0 + \alpha_2 A_1' B_0' + \alpha_2 A_0' B_1' + \alpha_2 A_1' B_1' \\
&= \alpha_2 (A_0' + A_1')(B_0' + B_1') \\
&= \alpha_2 \sum_{i,j=0}^{3} C_i D_j u_i \cdot v_j = \alpha_2 \sum_{i,j=0}^{3} C_{ij} u_i \cdot v_j.
\end{aligned}$$

◀

References

1. Farid Alizadeh. Interior point methods in semidefinite programming with applications to combinatorial optimization. *SIAM Journal on Optimization*, 5:13–51, 1993.
2. Anurag Anshu, David Gosset, and Karen Morenz. Beyond product state approximations for a quantum analogue of Max Cut. *arXiv:2003.14394 [quant-ph]*, March 2020. `arXiv:2003.14394`.
3. Sanjeev Arora, Eli Berger, Elad Hazan, Guy Kindler, and Muli Safra. On non-approximability for quadratic programs. In *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '05, pages 206–215, Washington, DC, USA, 2005. IEEE Computer Society. `doi:10.1109/SFCS.2005.57`.
4. Per Austrin. Towards sharp inapproximability for any 2-CSP. *SIAM Journal on Computing*, 39(6):2430–2463, 2010. `doi:10.1137/070711670`.
5. Nikhil Bansal, Sergey Bravyi, and Barbara M. Terhal. Classical approximation schemes for the ground-state energy of quantum and classical Ising spin Hamiltonians on planar graphs. Quant. Inf. Comp. Vol. 9, No.8, p. 0701 (2009), 2007. arXiv:0705.1115v4. `arXiv:0705.1115`.
6. Adam D. Bookatz. Qma-complete problems. *Quantum Info. Comput.*, 14(5&6):361–383, April 2014. URL: `http://dl.acm.org/citation.cfm?id=2638661.2638662`.
7. Fernando G. S. L. Brandão and Aram W. Harrow. Product-state approximations to quantum states. *Communications in Mathematical Physics*, 342(1):47–80, February 2016. `doi:10.1007/s00220-016-2575-1`.
8. Sergey Bravyi, David Gosset, Robert Koïig, and Kristan Temme. Approximation algorithms for quantum many-body problems. *Journal of Mathematical Physics*, 60(3):032203, 2019. `doi:10.1063/1.5085428`.
9. Jop Briët, Fernando Mário de Oliveira Filho, and Frank Vallentin. The positive semidefinite Grothendieck problem with rank constraint. In *Automata, Languages and Programming*, pages 31–42, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.
10. Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, FOCS '04, pages 54–60, Washington, DC, USA, 2004. IEEE Computer Society. `doi:10.1109/FOCS.2004.39`.
11. Toby Cubitt and Ashley Montanaro. Complexity classification of local hamiltonian problems. *SIAM Journal on Computing*, 45(2):268–316, 2016.
12. Sevag Gharibian and Julia Kempe. Approximation algorithms for QMA-complete problems. SIAM Journal on Computing 41(4): 1028-1050, 2012, 2011. `arXiv:1101.3884v1`. `doi:10.1137/110842272`.
13. Sevag Gharibian and Julia Kempe. *Hardness of approximation for quantum problems*, pages 387–398. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. `doi:10.1007/978-3-642-31594-7_33`.

14 Sevag Gharibian and Ojas Parekh. Almost optimal classical approximation algorithms for a quantum generalization of Max-Cut. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, 2019.

15 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.

16 Aram W. Harrow and Ashley Montanaro. Extremal eigenvalues of local Hamiltonians. *Quantum*, 1:6, April 2017. doi:10.22331/q-2017-04-25-6.

17 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, October 2004. doi:10.1109/FOCS.2004.49.

18 Subhash Khot and Assaf Naor. Grothendieck-type inequalities in combinatorial optimization. *Communications on Pure and Applied Mathematics*, 65(7):992–1035, July 2012. doi:10.1002/cpa.21398.

19 A. Yu. Kitaev, A. H. Shen, and M. N. Vyalyi. *Classical and Quantum Computation*. American Mathematical Society, Boston, MA, USA, 2002.

20 Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In *Integer Programming and Combinatorial Optimization*, pages 67–82, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg.

21 Konstantin Makarychev and Yury Makarychev. Approximation algorithms for CSPs. In *The Constraint Satisfaction Problem: Complexity and Approximability*, volume 7 of *Dagstuhl Follow-Ups*, pages 287–325. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2017. doi:10.4230/DFU.Vol7.15301.287.

22 Shiro Matuura and Tomomi Matsui. 0.863-approximation algorithm for MAX DICUT. In *Approximation, Randomization, and Combinatorial Optimization: Algorithms and Techniques*, pages 138–146, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

23 Uri Zwick. Analyzing the MAX 2-SAT and MAX DI-CUT approximation algorithms of Feige and Goemans, 2000.

Better and Simpler Learning-Augmented Online Caching

Alexander Wei
Harvard University, Cambridge, MA, USA
https://www.alexwei.org
weia@college.harvard.edu

Abstract

Lykouris and Vassilvitskii (ICML 2018) introduce a model of online caching with machine-learned advice that marries the predictive power of machine learning with the robustness guarantees of competitive analysis. In this model, each page request is augmented with a prediction for when that page will next be requested. The goal is to design algorithms that (1) perform well when the predictions are accurate and (2) are robust in the sense of worst-case competitive analysis.

We continue the study of algorithms for online caching with machine-learned advice, following the work of Lykouris and Vassilvitskii as well as Rohatgi (SODA 2020). Our main contribution is a substantially simpler algorithm that outperforms all existing approaches. This algorithm is a black-box combination of an algorithm that just naïvely follows the predictions with an optimal competitive algorithm for online caching. We further show that combining the naïve algorithm with LRU in a black-box manner is optimal among deterministic algorithms for this problem.

2012 ACM Subject Classification Theory of computation → Caching and paging algorithms; Theory of computation → Machine learning theory; Computing methodologies → Machine learning

Keywords and phrases Online caching, learning-augmented algorithms, beyond worst-case analysis

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.60

Category APPROX

Related Version A full version of the paper is available at https://arxiv.org/abs/2005.13716.

Acknowledgements I would like to thank Jelani Nelson for advising this project and Bailey Flanigan for providing many helpful references.

1 Introduction

The study of online algorithms traditionally focuses on *worst-case* robustness, where algorithms provide the same competitive guarantee against the offline optimal over all possible inputs. In recent years, however, there has been a surge of interest in online algorithms for structured inputs [20, 18, 24, 11, 15, 16, 25, 17, 23]. A principal motivation for these works is the philosophy of "beyond worst-case analysis" [13, 26]: Many practical settings have inputs that follow restricted patterns, making worst-case competitive analysis too pessimistic to inform practice. Algorithms designed with the worst case in mind can be hamstrung by these considerations, sacrificing performance on "easy" inputs in order to eke out a better worst-case guarantee.

Learning-augmented online algorithms, introduced by Lykouris and Vassilvitskii [18] and Purohit et al. [24], is a "beyond worst-case" framework motivated by the powerful predictive abilities of modern machine learning. In this framework, the classical online algorithm model is augmented with a machine-learned oracle that predicts future inputs. A concern with simply relying on the oracle is that machine learning models typically have few worst-case guarantees. Thus, with learning-augmented online algorithms, we seek to obtain the best of both worlds. Given a predictor, our objective is to design algorithms that:


© Alexander Wei;
licensed under Creative Commons License CC-BY



1. Perform well in the optimistic scenario, where the predictor has low error;
2. Remain robust in the classical worst-case sense, when the predictor can be arbitrarily bad.

In other words, we want our algorithm to be $c(\eta)$-competitive against the offline optimal on all inputs, for c a function of the predictor's total error η, such that $\sup_\eta c(\eta) \leq \gamma$ for some constant $\gamma \geq 1$. This γ is the classical worst-case competitive ratio and is a measure of the robustness of our algorithm.

This paper focuses on learning-augmented *online caching* [18, 25]. In the online caching (a.k.a. online paging) problem, one maintains a cache of size k online while serving requests for pages that may or may not be in the cache. For simplicity, assume that pages must always be served from the cache and that bringing a page into the cache has unit cost. (In particular, if the cache is full, bringing a page into the cache requires also *evicting* a page already in the cache.) Thus, one wishes to minimize the number of *cache misses*, i.e., requests for which the page is not already in the cache. This is a classical online problem that has been the subject of extensive study over the past several decades (see [6] for an overview). From the worst-case perspective, this problem is well-understood for not only the version stated above [27, 10, 1], but also for weighted generalizations [3, 4].

Online caching in the learning-augmented context was first considered by Lykouris and Vassilvitskii [18]. They introduce a model of prediction where the predictor, upon the arrival of each page, predicts the next time that this page will be requested. They show that the BlindOracle algorithm, which follows the predictor naïvely and evicts the page with the latest predicted arrival time, can have unbounded competitive ratio (i.e., is *non-robust*). They then give a different algorithm, PredictiveMarker, based on the Marker algorithm of Fiat et al. [10], that achieves a competitive ratio of

$$2 + O\left(\min\left(\sqrt{\frac{\eta}{\mathsf{OPT}}}, \log k\right)\right),$$

where η is the ℓ_1 error of the predictor and OPT is the cost of the offline optimal. (For precise definitions, refer to Section 2.1.) Notably, this competitive ratio approaches 2 as the error η goes to 0 and is bounded by $O(\log k)$ regardless of how large η gets.

In recent work, Rohatgi [25] introduces the LNonMarker algorithm, which is also based on randomized marking, and shows that LNonMarker achieves a competitive ratio of

$$O\left(1 + \min\left(\frac{\log k}{k}\frac{\eta}{\mathsf{OPT}}, \log k\right)\right).$$

This bound is obtained by constructing a non-robust algorithm and then using the black-box combination technique discussed in [18] to combine this non-robust algorithm with the Marker algorithm. Rohatgi also shows a lower bound of

$$1 + \Omega\left(\min\left(\log\left(\frac{1}{k \log k}\frac{\eta}{\mathsf{OPT}}\right), \log k\right)\right)$$

for the competitive ratio of any learning-augmented online algorithm for caching in terms of k, OPT, and η. The main difference between two bounds is that the former has a linear dependence on η/OPT, whereas the latter has only a logarithmic dependence on the same. We will make some progress in closing this gap and also show that deterministic algorithms fundamentally cannot approach Rohatgi's lower bound.

1.1 Our Contribution

We show that the strikingly simple approach of combining BlindOracle with an $O(\log k)$-competitive online caching algorithm (e.g., Marker) in a black-box fashion achieves state-of-the-art performance, improving over LNonMarker with a competitive ratio bound of

$$O\left(1+\min\left(\frac{1}{k}\frac{\eta}{\mathsf{OPT}}, \log k\right)\right).$$

Thus, although BlindOracle was previously shown to be non-robust [18], our result demonstrates that using BlindOracle appropriately can actually lead to very effective algorithms for learning-augmented online caching.

In addition to achieving better theoretical bounds, our approach to learning-augmented online caching is substantially simpler than previous work. The algorithms of Lykouris and Vassilvitskii [18] and Rohatgi [25] rely on intricate constructions based on randomized marking, whereas our main ingredient is a careful analysis of perhaps the simplest algorithm possible. We thus believe that our approach may yield better practical performance and may generalize more readily to other learning-augmented settings. We note that our optimal deterministic algorithm is especially simple: For each eviction, "follow" whichever of BlindOracle and LRU has performed better so far.

The crux of our result is a tight analysis of BlindOracle's performance as a function of η. We prove that BlindOracle has excellent performance when η/OPT is small, improving over $O(\log k)$-competitive online caching algorithms for η up to $O(k \log k) \cdot \mathsf{OPT}$. This is in contrast to the lower bound on BlindOracle given by Lykouris and Vassilvitskii [18], who rule out BlindOracle due to its poor performance when $k = 2$. In particular, our bound has a $1/k$ dependence on k, meaning BlindOracle obtains drastically better bounds for larger k. Stated formally, our main theorem is the following:

▶ **Theorem 1.1.** *For learning-augmented online caching, BlindOracle obtains a competitive ratio of*

$$\min\left(1+2\frac{\eta}{\mathsf{OPT}}, 4+\frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}\right),$$

where η is the ℓ_1 loss incurred by the predictor and OPT is the optimal offline cost. (For precise definitions, see Section 2.1.)

We then obtain robust deterministic and randomized algorithms for learning-augmented online caching as corollaries, by combining BlindOracle with LRU and Equitable [1]. We apply the algorithms of Fiat et al. [10] and Blum and Burch [5] for combining online algorithms in a black-box manner online. These algorithms achieve better constants in the competitive ratio than the approach discussed by Lykouris and Vassilvitskii [18] and applied by Rohatgi [25]. By composing our analysis of BlindOracle with these "combiners," we obtain constants in the competitive ratio that are significantly lower than those of previous works. Indeed, we have the following corollaries for deterministic and randomized algorithms for learning-augmented online caching:

▶ **Corollary 1.2.** *There exists a deterministic algorithm for learning-augmented online caching that achieves a competitive ratio of*

$$2\min\left(\min\left(1+2\frac{\eta}{\mathsf{OPT}}, 4+\frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}\right), k\right).$$

▶ **Corollary 1.3.** *There exists a randomized algorithm for learning-augmented online caching that achieves a competitive ratio of*

$$(1+\gamma)\min\left(\min\left(1+2\frac{\eta}{\mathsf{OPT}}, 4+\frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}\right), H_k\right)$$

for any $\gamma \in (0, 1/4)$.[1] *(Here,* $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k} = \ln(k) + O(1)$ *is the* k*-th harmonic number.)*

Finally, we show that combining BlindOracle with a k-competitive deterministic algorithm (e.g., LRU [27]) is the best one could hope to do among deterministic algorithms for learning-augmented online caching. In particular, we show that a linear dependence on $\eta/(k \cdot \mathsf{OPT})$ in the competitive ratio is necessary. Therefore, if a logarithmic dependence on $\eta/(k \cdot \mathsf{OPT})$ is to be achieved, as in Rohatgi's lower bound, then randomization is needed, (perhaps surprisingly) even in the regime where $\eta/(k \cdot \mathsf{OPT})$ is bounded.

▶ **Theorem 1.4.** *The competitive ratio bound for any deterministic learning-augmented online caching algorithm must be at least*

$$1 + \Omega\left(\min\left(\frac{1}{k}\frac{\eta}{\mathsf{OPT}}, k\right)\right)$$

in terms of k *and* η/OPT.

1.2 Related Work

In addition to the predecessor works by Lykouris and Vassilvitskii [18] and Rohatgi [25] on learning-augmented online caching, there have been several other recent papers in the space of learning-augmented online algorithms: Medina and Vassilvitskii [20] study repeated posted-price auctions, Purohit et al. [24] and Gollapudi and Panigrahi [11] study the ski rental problem, and Purohit et al. [24], Lattanzi et al. [17], and Mitzenmacher [23] study online scheduling. Of these, the scheduling algorithm of Purohit et al. [24] is the most similar in spirit to this present work: Both algorithms are based on combining a naïve and optimistic algorithm with a robust algorithm.

Other threads of research falling under beyond worst-case online algorithms include work on combining multiple algorithms with different performance characteristics [10, 5, 19, 11], designing online algorithms with distributional assumptions (e.g., stochasticity) on the input [13, 8, 21], and semi-online algorithms, where the input is assumed to have a predictable offline component and an adversarial online component [15, 16].

The idea of learning-augmentation has also been explored in many other algorithmic and data structural settings in recent years. These include learned indices [14], bloom filters [22], frequency estimation in streams [12], and nearest neighbor search [9], among others.

Finally, advice for online algorithms has also been considered with a more complexity theoretic spirit through the study of advice complexity of online algorithms; see the survey of Boyar et al. [7] for an overview.

1.3 Recent Developments

Recently, in work done independently of and concurrently with this paper, Antoniadis et al. [2] also study a BlindOracle-like algorithm, which they term FollowThePrediction, in the more general setting of learning-augmented metrical task systems; they also use the "combiner"

[1] The trade-off in γ and the cost is additive; thus, it does not factor into the competitive ratio.

of Blum and Burch [5] to make this algorithm robust. However, their prediction model, when specialized to online caching, is incomparable to that of Lykouris and Vassilvitskii [18] (which we follow).[2] Thus, the theoretical results proved in these two models do not imply each other.

1.4 Outline

The remainder of this paper is organized as follows: In Section 2, we formally describe our model of learning-augmented online caching and BlindOracle; we also introduce background results that will later be used in our proofs. In Section 3, we prove our upper bounds (i.e., Theorem 1.1 and Corollaries 1.2 and 1.3). In Section 4, we prove our lower bound (Theorem 1.4) for deterministic algorithms. Section 5 concludes.

2 Preliminaries

2.1 Setup and Notation

In the online caching problem, we receive a sequence $\sigma = (\sigma_1, \ldots, \sigma_n)$ of page requests online, and our goal is to serve these requests using a cache of size k while minimizing cost. In this problem, pages must be served from the cache and can be served at no cost; however, evicting a page from the cache has unit cost.[3]

We will establish competitive bounds comparing the performance of two caching algorithms $\mathcal{A}$ and $\mathcal{B}$. More precisely, we will show bounds of the form

$$\mathsf{ALG}_{\mathcal{B}}(\sigma) \leq \gamma \cdot \mathsf{ALG}_{\mathcal{A}}(\sigma) + O(1),$$

where $\mathsf{ALG}_{\mathcal{A}}(\sigma)$ and $\mathsf{ALG}_{\mathcal{B}}(\sigma)$ are the costs of $\mathcal{A}$ and $\mathcal{B}$, respectively, as measured in number of evictions made while serving a sequence σ of page requests. Ultimately, $\mathcal{A}$ will be the optimal offline algorithm and $\mathcal{B}$ will be BlindOracle. We will also use $\mathsf{OPT}(\sigma)$ to denote the optimal offline cost; that is, $\mathsf{OPT}(\sigma)$ is the minimum possible cost of serving the request sequence σ. We will omit the argument σ when the context is clear (i.e., just writing $\mathsf{ALG}_{\mathcal{A}}$ to represent $\mathsf{ALG}_{\mathcal{A}}(\sigma)$).

In our analysis, we use A_t and B_t to denote the cache states of $\mathcal{A}$ and $\mathcal{B}$, respectively, just before the t-th request. Formally, A_t and B_t are subsets of $\{1, \ldots, t-1\}$ of size at most k, containing for each cached page the index at which it was last served. That is, when serving the t-th request, we remove some old request index t' from the cache and insert t. Thus, if t' is such that $\sigma_t = \sigma_{t'}$, this operation is free; otherwise, it has unit cost. In the sequel, we will also refer to these indices t as *page requests*.

In the learning-augmented online caching problem, the t-th page request comes with a prediction h_t for the next time page σ_t is requested. That is, at the time of the t-th request, our algorithm receives the pair (σ_t, h_t). Let $h = (h_1, \ldots, h_n)$ be the tuple of all n predictions. To define a notion of loss, let y_t denote for each t the next time page t is actually requested, with $y_t = n + 1$ if page σ_t is never requested again. The ℓ_1 loss is then defined to be

$$\eta(\sigma, h) = \sum_t |h_t - y_t|.$$

[2] Namely, their algorithms expect predictions to be in a different form: They expect predictions to be cache states (i.e., the set of pages in the cache at time t) rather than next arrival times of pages. Moreover, there exist sequences of "corresponding" inputs for each of these two models such that the predictor error approaches infinity in one model while remaining constant in the other.

[3] Observe that this formulation is equivalent to the "standard" formulation of online caching, where each cache miss has unit cost, up to an additive constant of k.

We will omit arguments to η if the context is clear. Note that if $\eta(\sigma, h) = 0$, then the offline optimal can be obtained, as the optimal algorithm always evicts the page that is next requested furthest into the future.

In stating our bounds, the essential quantity is often η/OPT. To make this clear, we will take $\varepsilon = \eta/\mathsf{OPT}$ and state our bounds in terms of ε in the sequel.

2.1.1 Inversions

Call a pair (i, j) of page requests an *inversion* if $y_i < y_j$ but $h_i \geq h_j$. Let $M(\sigma, h)$ denote the total number of inversions between the pair of sequences σ and h. We will omit arguments to M when the context is clear.

2.1.2 BlindOracle

We formally define the BlindOracle algorithm as follows: For each page request, if the requested page is already in the cache, do nothing. Otherwise, evict the page request p whose predicted next arrival time h_p is furthest into the future, with ties broken consistently (e.g., by always evicting the least recently used page among those with maximal h_p).

2.2 Combining Online Algorithms Competitively

To define our algorithms, we will need two classical results on competitively combining online algorithms online, due to Fiat et al. [10] and Blum and Burch [5], respectively. This type of "black-box" combination was also considered by Lykouris and Vassilvitskii [18], but their approach has a worse constant than that of Fiat et al. [10]. We also note that results of a similar flavor are proven by Purohit et al. [24] and Mahdian et al. [19], but for other online problems.

The question of combining multiple online algorithms while remaining competitive against each was first considered in the seminal paper of Fiat et al. [10]. They consider combining n online algorithms $\mathcal{B}_1, \ldots, \mathcal{B}_n$ for the online caching problem into a single algorithm $\mathcal{B}$ such that $\mathcal{B}$ is C_i-competitive against $\mathcal{B}_i$ for each i. They show such an $\mathcal{B}$ is achievable if and only if

$$\sum_{i=1}^{n} \frac{1}{C_i} \leq 1.$$

We will need only the special case of $n = 2$ and $C_1 = C_2 = 2$, which we state below:

▶ **Theorem 2.1** ([10], special case)**.** *Given two algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ for the online caching problem, there exists an algorithm $\mathcal{B}$ such that*

$$\mathsf{ALG}_{\mathcal{B}}(\sigma) \leq 2\min(\mathsf{ALG}_{\mathcal{B}_1}(\sigma), \mathsf{ALG}_{\mathcal{B}_2}(\sigma)) + O(1).$$

Moreover, if $\mathcal{B}_1$ and $\mathcal{B}_2$ are deterministic, then so is $\mathcal{B}$.

We note that this can be done deterministically with a "follow-the-leader" approach, where we simulate both algorithms and at each step evict any page that is not in the cache of the better performing algorithm (as measured by total number of evictions after serving the current request).

Blum and Burch [5] show that one can obtain a better approximation factor using a *randomized* scheme, namely multiplicative weights.[4] That is, at each point in time, the probability that the combined algorithm is following one of the n algorithms is given by a probability distribution over the n algorithms governed by the multiplicative weights update rule. For $n = 2$, their result can be stated as follows:

▶ **Theorem 2.2** ([5], special case). *Given two algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ for the online caching problem and any γ such that $0 < \gamma < 1/4$, there exists an algorithm $\mathcal{B}$ such that*

$$\mathsf{ALG}_{\mathcal{B}}(\sigma) \leq (1+\gamma)\min(\mathsf{ALG}_{\mathcal{B}_1}(\sigma), \mathsf{ALG}_{\mathcal{B}_2}(\sigma)) + O(\gamma^{-1}k).$$

▶ Remark. Although we do not state the versions of these results for combining several algorithms, one can imagine that they could be useful if one wishes to ensemble multiple machine-learned predictors.

2.3 From ℓ_1 Loss to Inversions

We now state a lemma of Rohatgi [25] that relates ℓ_1 loss to the number of inversions, letting us lower bound the ℓ_1 loss $\eta(\sigma, h)$ by lower bounding the number of inversions $M(\sigma, h)$. Thus, instead of reasoning in terms of ℓ_1 loss, we will reason in terms of inversions.

▶ **Lemma 2.3** ([25, Lemma 11]). *For any σ and h, $\eta(\sigma, h) \geq \frac{1}{2}M(\sigma, h)$.*

With this lemma, it suffices (up to a factor of 2) to give our competitive ratio upper bounds in terms of the number of inversions M.

3 Upper Bounds

3.1 A First Analysis of BlindOracle

In this section, we give a first analysis of BlindOracle, showing that it gets very good performance when the ratio $\varepsilon = \eta/\mathsf{OPT}$ is very small. In particular, our analysis shows that as $\varepsilon \to 0$, the competitive ratio achieved approaches 1.

Let $\mathcal{A}$ be the offline optimal algorithm (i.e., such that $\mathsf{ALG}_{\mathcal{A}} = \mathsf{OPT}$). Let $\mathcal{B}$ be BlindOracle. Note that we can think of each of $\mathsf{ALG}_{\mathcal{A}}$, $\mathsf{ALG}_{\mathcal{B}}$, and M as functions of the time t, i.e., they are the cost of $\mathcal{A}$, the cost of $\mathcal{B}$, and the number of inversions, respectively, on the prefix consisting of the first $t-1$ requests.[5] We use the Δ operator to denote the change (in a function of t) from time t to time $t+1$. For example, $\Delta\mathsf{ALG}_{\mathcal{A}} = 1$ if $\mathsf{ALG}_{\mathcal{A}}$ evicts an element upon the t-th request.

In our analysis, we maintain a matching X_t between A_t and B_t at all times t. Call a matching *valid* if it consists only of pairs $(a, b) \in A_t \times B_t$ such that the next arrival of b is no later than the next arrival of a. Our matching $X_t \subseteq A_t \times B_t$ will be valid throughout the execution of the algorithm.

We now proceed with a potential function analysis, taking our potential Φ (as a function of A_t, B_t, and X_t) to be the number of unmatched pages in B_t. For notational simplicity, we denote $\Phi(A_t, B_t, X_t)$ by $\Phi(t)$. Given this setup, we show:

[4] The result of Blum and Burch [5] in fact holds more generally for all metrical task systems.
[5] This indexing is to be consistent with the definitions of A_t and B_t.

▶ **Proposition 3.1.** *There exists a valid matching X_n such that*

$$\mathsf{ALG}_{\mathcal{B}} + \Phi(n) \leq \mathsf{OPT} + M.$$

Proof. We induct on the length n of the input and perform a case analysis to show that we can maintain a valid matching X_t such that at each time step, the right-hand side increases at least as much as the left-hand side, i.e., $\Delta\mathsf{ALG}_{\mathcal{B}} + \Delta\Phi \leq \Delta\mathsf{OPT} + \Delta M$.

For our base case, note that $A_1 = B_1$, so we may take X_1 to be the identity matching.

Now, upon a request at time t, we update X_t according to the following cases (and with the consequences listed for each case):

1. The requested page p is in both A_t and B_t.
   a. The cached pages are matched to each other.
      - Do nothing.
   b. Otherwise:
      i. Both cached pages are matched.
         - Remove the pairs (c, p) and (p, d) from X_t.
         - Add the pairs (p, p) and (c, d) to X_t.
         - As a result:
            - Nothing changes. (For conciseness, we omit obviously unchanged quantities here and for the remainder of this proof.)
      ii. Otherwise:
         - Remove any pairs involving p from X_t. (There is at most one such pair.)
         - Add the pair (p, p) to X_t.
         - As a result:
            - $\Delta\Phi \leq 0$.
2. The requested page p is in B_t only.
   - Remove any pairs involving the evicted page a from X_t. (There is at most one such pair.)
   - Remove any pairs involving the requested page p from X_t. (There is at most one such pair.)
   - Add the pair (p, p) to X_t.
   - As a result:
      - $\Delta\mathsf{OPT} = 1$.
      - $\Delta\Phi \leq 1$.
3. The requested page p is in A_t only.
   a. The evicted page $b \in B_t$ is unmatched.
      - Add the pair (p, p) to X_t. (The arriving page $p \in A_t$ cannot be in any valid matching.)
      - As a result:
         - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
         - $\Delta\Phi = -1$.
   b. The evicted page $b \in B_t$ is matched.
      i. b arrives later than all unmatched pages in B_t.
         - Remove the pair (c, b) involving the evicted page b from X_t.
         - Add the pair (c, b') to X_t, where $b' \in B_t$ is any unmatched page.
         - Add the pair (p, p) to X_t. (The arriving page $p \in A_t$ cannot be in any valid matching.)
         - As a result:
            - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
            - $\Delta\Phi = -1$.

   - **ii.** There is an unmatched page $b' \in B_t$ arriving later than b.
     - Remove the pair (c, b) involving the evicted page b from X_t.
     - Add the pair (p, p) to X_t. (The arriving page $p \in A_t$ cannot be in any valid matching.)
     - As a result:
       - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
       - $\Delta\Phi = 0$.
       - $\Delta M = 1$, as there is an inversion between b and b'. (Note that we do not count this inversion ever again, as b gets evicted.)

4. The requested page p is in neither A_t nor B_t.
   - **a.** $\mathcal{A}$ evicts an unmatched page $a \in A_t$.
     - **i.** $\mathcal{B}$ evicts an unmatched page $b \in B_t$.
       - Add the pair (p, p) to X_t.
       - As a result:
         - $\Delta\mathsf{OPT} = 1$.
         - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
         - $\Delta\Phi = 1$.
     - **ii.** $\mathcal{B}$ evicts a matched page $b \in B_t$.
       - Remove the pair (c, b) involving b from X_t.
       - Add the pair (p, p) to X_t.
       - As a result:
         - $\Delta\mathsf{OPT} = 1$.
         - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
   - **b.** $\mathcal{A}$ evicts a matched page $a \in A_t$.
     - **i.** $\mathcal{B}$ evicts an unmatched page $b \in B_t$.
       - Remove the pair (a, d) involving a from X_t.
       - Add the pair (p, p) to X_t.
       - As a result:
         - $\Delta\mathsf{OPT} = 1$.
         - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
     - **ii.** $\mathcal{B}$ evicts a matched page $b \in B_t$.
       - Remove the pair (a, d) involving a from X_t.
       - Remove the pair (c, b) involving b from X_t.
       - Add the pair (p, p) to X_t.
       - As a result:
         - $\Delta\mathsf{OPT} = 1$.
         - $\Delta\mathsf{ALG}_{\mathcal{B}} = 1$.
         - Note that either b arrives after d, in which case we can add (c, d) to X_t and $\Delta\Phi = 0$, or the pair (b, d) forms an inversion, in which case $\Delta\Phi = 1$ and $\Delta M = 1$. (As before, since b is getting evicted, we will not count this pair twice.)

It is not hard to verify that the change in the left-hand side of the bound is no more than the change in the right-hand side in each of the cases listed above, from which the proposition follows. ◀

▶ **Proposition 3.2.** *The competitive ratio of algorithm $\mathcal{B}$ is at most $1 + 2\varepsilon$.*

Proof. Note that 2η is bounded below by the number of inversions M of (σ, h) by Lemma 2.3. By Proposition 3.1, $\mathsf{ALG}_{\mathcal{A}} \leq \mathsf{OPT} + M$, so $\mathsf{ALG}_{\mathcal{A}}/\mathsf{OPT} \leq 1 + M/\mathsf{OPT} \leq 1 + 2\varepsilon$. ◀

3.2 A More Careful Analysis

We now give an asymptotically better (in k) bound for the performance of BlindOracle. A more careful analysis is needed to show an upper bound with a $1/k$ coefficient on the ratio $\varepsilon = \eta/\mathsf{OPT}$. We use the same high-level approach for the proof as before, but with a more complicated potential function. Again, $\mathcal{A}$ is the offline optimal algorithm and $\mathcal{B}$ is the BlindOracle algorithm, and also as before, we use Δ to denote change (in functions of t) from request t to request $t+1$.

We maintain in this proof a matching X_t over pairs of page requests $(a, b) \in A_t \times B_t$ such that $h_a \geq h_b$ for each time step t. (Recall that h_a is the prediction for the next time that page σ_a is requested.) We also require that X_t restricts to the identity matching on $A_t \cap B_t$. That is, if a page p is in both caches at time t, then it is matched to itself in X_t. If X_t satisfies both of these properties, we say that it is *allowable*. Our potential function Φ – which will really be a sum of three separate potential functions – will be a function of A_t, B_t, and an allowable matching X_t. For notational simplicity, we denote $\Phi(A_t, B_t, X_t)$ by $\Phi(t)$.

Given A_t, B_t, and X_t at time t, define $\Phi_0(t)$ to be the number of $b \in B_t$ that are unmatched. Define $\Phi_1(t)$ to be the number of $b \in B_t$ such that $(b, b) \notin X_t$. In other words, Φ_1 counts how many page requests in B_t are not matched to the *same* page request in A_t. For a page request p, let $z_p(t)$ be the number of pages in B_t predicted to appear before h_p (with pages predicted to appear at the same time tie-broken in a consistent manner, e.g., by the last time each page was requested[6]). Next, define

$$\Phi_2(t) = \sum_{(a,b)\in X_t} \big(z_b(t) - z_a(t)\big).$$

With these "sub"-potential functions defined, we take

$$\Phi(t) = (k-1)\Phi_0(t) + (k-1)\Phi_1(t) + \Phi_2(t)$$

as our overall potential function. This completes our setup for the analysis. But before we give the details of the proof, we make a few observations about these definitions and discuss some intuition.

First, observe that for any p, we have $0 \leq z_p(t) \leq k$. And if $p \in B_t$, then $z_p(t) \leq k-1$ because there are only $k-1$ other pages that could be predicted to appear before. Next, if $(a, b) \in X_t$, then $z_a(t) \geq z_b(t)$ as $h_a \geq h_b$. Hence $\Phi_2(t) \leq 0$ always. As a result, Φ may sometimes be negative, but it satisfies $|\Phi(t)| \leq 3k^2$ for all t. Finally, we note that Φ_2 is equivalently a sum over the $(a, b) \in X_t$ such that $a \neq b$.

In terms of intuition for these definitions, the main substance perhaps lies in Φ_2: Suppose $(a, b) \in X_t$ and $a \neq b$; in particular, $a \notin B_t$. The quantity $z_a(t) - z_b(t)$ is the future number of inversions that we are "guaranteed" from this pair (a, b). If we request σ_p for any p such that $h_b < h_p < h_a$, then we have an inversion (p, b). On the other hand, if σ_p is not requested before the next time σ_a is requested, then the request for σ_a forms an inversion (a, p). Thus, the quantity $M - \Phi_2$ accounts not only for the inversions already encountered, but also for future inversions. The matching of a and b also serves another purpose: We couple the decrease of z_a to the decrease of z_b whenever z_a decreases without generating an inversion. Therefore, if z_a is small when σ_a is next requested, then $z_b \leq z_a$ is also small. It follows that if we form a new pair (a', b), then $z_{a'} - z_b$ will be comparatively large, i.e., we get many future inversions.

[6] Alternatively, one can simply perturb all predictions by infinitesimal amounts so that there are no ties.

We execute on this intuition to prove the following proposition:

▶ **Proposition 3.3.** *For any input (σ, h), there exists a matching $X_n \subseteq A_n \times B_n$ consisting only of pairs (a, b) satisfying $h_a \geq h_b$ such that*

$$(k-1)\mathsf{ALG}_{\mathcal{B}} + \Phi(n) \leq 2k \cdot \mathsf{OPT} + 2M.$$

Proof. Like in the proof of Proposition 3.1, we induct on the length of the input and perform a case analysis to show that we can maintain a allowable matching X_t such that the right-hand side increases at least as much as the left-hand side for each page request.

We split the serving of each page request into two phases:

1. **Matching.** Update X_t so that the page requests in A_t and B_t that are to be removed are unmatched. (Note that page requests are removed either because the corresponding page was requested again or because the corresponding page was evicted.)
2. **Updating.** Replace a page request from each of A_t and B_t with the new request and add the new page request pair (t, t) into X_t.

In doing so, we ensure that the pages to be removed are unmatched by the time updating occurs, so they do not contribute to Φ_2.

We first analyze how updating affects the overall potential Φ. This operation decreases each of Φ_0 and Φ_1 by 1, since we remove an unmatched pair and replace it with the matched pair (t, t). For Φ_2, observe that for a matched pair (a, b), the difference $z_b - z_a$ increases on the t-th request only if a page predicted to arrive between b and p is removed. This occurs if and only if there is a $p \in B_t$ such that $\sigma_p = \sigma_t$ and $h_b < h_p < h_a$. In this case, the pair (p, b) forms an inversion, i.e., it increases M by 1. Any inversion (p, b) is counted at most once this way because p is getting removed from B_t. Thus $\Delta\Phi_2 \leq \Delta M$. In summary, updating leaves us with an extra $2(k-1)$ in potential due to the changes in Φ_0 and Φ_1; the change in potential of Φ_2 is offset by the change in one of the two M terms on the right-hand side.

We now specify the matching phase in greater detail and analyze it using a case analysis. We will show

$$(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} + \Delta\Phi \leq 2k \cdot \Delta\mathsf{OPT} + \Delta M + 2(k-1),$$

with the extra $2(k-1)$ being covered by the updating step. The analysis proceeds as follows:

1. The requested page is in both A_t and B_t.
   - The previous page requests for σ_t in A_t and B_t are matched to each other by assumption, so we can just unmatch them.
   - As a result:
     - $(k-1)\Delta\Phi_0 = k-1$.
     - $(k-1)\Delta\Phi_1 = k-1$.
2. The requested page is in A_t only.
   a. The previous request $p \in A_t$ for the requested page is matched as (p, d) and the page request $b \in B_t$ evicted by $\mathcal{B}$ is matched as (c, b).
      - Unmatch (p, d) and (c, b) and match (c, d). The latter is allowable because $h_c \geq h_b \geq h_d$.
      - As a result:
        - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.
        - $(k-1)\Delta\Phi_0 = k-1$.
        - $(k-1)\Delta\Phi_1 \leq k-1$, since $p \neq d$.
        - $\Delta\Phi_2 = z_p - (k-1)$, since $z_b = k-1$.
        - $\Delta M \geq z_p$, since the arrival of σ_p generates z_p inversions of the form (p, b') for all $b' \in B_t$ such that $h_p > h_{b'}$.

b. The previous request $p \in A_t$ for the requested page is matched as (p, d) and the page request $b \in B_t$ evicted by $\mathcal{B}$ is unmatched.
- Unmatch (p, d).
- As a result:
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.
  - $(k-1)\Delta\Phi_0 = k-1$.
  - $(k-1)\Delta\Phi_1 = 0$, since $p \neq d$.
  - $\Delta\Phi_2 = z_p - z_b \leq z_p$.
  - $\Delta M \geq z_p$, since the arrival of σ_p generates z_p inversions of the form (p, b') for all $b' \in B_t$ such that $h_p \geq h_{b'}$.

c. The previous request $p \in A_t$ for the requested page is unmatched and the page request $b \in B_t$ evicted by $\mathcal{B}$ is matched as (c, b).
- Unmatch (c, b) and match (c, d) for an arbitrary unmatched $d \in B_t \setminus \{b\}$. This is allowable because $h_c \geq h_b \geq h_d$.
- As a result:
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.
  - $(k-1)\Delta\Phi_0 = 0$.
  - $(k-1)\Delta\Phi_1 \leq k-1$.
  - $\Delta\Phi_2 = z_d - (k-1) \leq 0$, since $z_b = k-1$.

d. The previous request $p \in A_t$ for the requested page is unmatched and the page request $b \in B_t$ evicted by $\mathcal{B}$ is unmatched.
- Do nothing.
- As a result:
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.

3. The requested page is in B_t only.

a. The previous request $p \in B_t$ for the requested page is matched as (c, p) and the page request $a \in A_t$ evicted by $\mathcal{A}$ is matched as (a, d).
- Unmatch (c, p) and (a, d).
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\Phi_0 = 2(k-1)$.
  - If $a = d$:
    - $(k-1)\Delta\Phi_1 = k-1$, since $c \neq p$.
    - $\Delta\Phi_2 = z_c - z_p \leq k$, since $z_c \leq k$.
  - Otherwise, if $a \neq d$:
    - $(k-1)\Delta\Phi_1 = 0$, since $c \neq p$.
    - $\Delta\Phi_2 = (z_c - z_p) + (z_a - z_d) \leq 2k$, since $z_c, z_a \leq k$.

b. The previous request $p \in B_t$ for the requested page is matched as (c, p) and the page request $a \in A_t$ evicted by $\mathcal{A}$ is unmatched.
- Unmatch (c, p).
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\Phi_0 = k-1$.
  - $(k-1)\Delta\Phi_1 = 0$, since $c \neq p$.
  - $\Delta\Phi_2 \leq z_c - z_p \leq k$, since $z_c \leq k$.

c. The previous request $p \in B_t$ for the requested page is unmatched and the page request $a \in A_t$ evicted by $\mathcal{A}$ is matched as (a, d).
- Unmatch (a, d).

- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\Phi_0 = k-1$.
  - $(k-1)\Delta\Phi_1 \leq k-1$.
  - $\Delta\Phi_2 \leq z_a - z_d \leq k$, since $z_a \leq k$.

d. The previous request $p \in B_t$ for the requested page is unmatched and the page request $a \in A_t$ evicted by $\mathcal{A}$ is unmatched.

- Do nothing.
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.

4. The requested page is in neither A_t nor B_t.

a. The previous request $a \in A_t$ evicted by $\mathcal{A}$ is matched as (a, d) and the page request $b \in B_t$ evicted by $\mathcal{B}$ is matched as (c, b).

- If $(a, d) = (c, b)$, simply unmatch (a, d) (and thus (c, b)). Otherwise, unmatch (a, d) and (c, b) and match (c, d). The latter is allowable because $h_c \geq h_b \geq h_d$.
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.
  - $(k-1)\Delta\Phi_0 = k-1$.
  - $(k-1)\Delta\Phi_1 \leq 2(k-1)$.
  - $\Delta\Phi_2 = z_a - (k-1) \leq 0$, since $z_b = k-1$.

b. The previous request $a \in A_t$ evicted by $\mathcal{A}$ is matched as (a, d) and the page request $b \in B_t$ evicted by $\mathcal{B}$ is unmatched.

- Unmatch (a, d).
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.
  - $(k-1)\Delta\Phi_0 = k-1$.
  - $(k-1)\Delta\Phi_1 \leq k-1$.
  - $\Delta\Phi_2 = z_a - z_d \leq k$, since $z_a \leq k$.

c. The previous request $a \in A_t$ evicted by $\mathcal{A}$ is unmatched and the page request $b \in B_t$ evicted by $\mathcal{B}$ is matched as (c, b).

- Unmatch (c, b) and match (c, d) for an arbitrary unmatched $d \in B_t \setminus \{b\}$. This is allowable because $h_c \geq h_b \geq h_d$.
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.
  - $(k-1)\Delta\Phi_0 = 0$.
  - $(k-1)\Delta\Phi_1 \leq k-1$.
  - $\Delta\Phi_2 = z_d - (k-1) \leq 0$, since $z_b = k-1$.

d. The previous request $a \in A_t$ evicted by $\mathcal{A}$ is unmatched and the page request $b \in B_t$ evicted by $\mathcal{B}$ is unmatched.

- Do nothing.
- As a result:
  - $2k \cdot \Delta\mathsf{OPT} = 2k$.
  - $(k-1)\Delta\mathsf{ALG}_{\mathcal{B}} = k-1$.

Recall that we have an extra $2(k-1)$ in potential from the updating phase that we can use to defray the costs of the matching phase. One can verify that this is sufficient for all of the cases described above – the "tight" cases are 1, 2(a), 2(b), 2(c), 3(a), and 4(a). The proposition now follows. ◀

▶ **Proposition 3.4.** *The competitive ratio of algorithm $\mathcal{B}$ is at most $2+2/(k-1)+4\varepsilon/(k-1)$.*

Proof. Compose Proposition 3.3 with Lemma 2.3. In particular, we have

$$\frac{\mathsf{ALG}_{\mathcal{B}}}{\mathsf{OPT}} + \frac{O(k)}{\mathsf{OPT}} \leq \frac{2k}{k-1} + \frac{2M}{(k-1)\mathsf{OPT}} \leq 2 + \frac{2}{k-1} + \frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}.$$ ◀

3.3 Finishing Up

The results stated in Section 1 now follow easily from what we have already shown:

▶ **Theorem 1.1.** *For learning-augmented online caching, BlindOracle obtains a competitive ratio of*

$$\min\left(1 + 2\frac{\eta}{\mathsf{OPT}}, 4 + \frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}\right),$$

where η is the ℓ_1 loss incurred by the predictor and OPT is the optimal offline cost. (For precise definitions, see Section 2.1.)

Proof. From the analysis of the previous two sections, the desired bound immediately follows from taking the minimum of the bounds in Propositions 3.2 and 3.4, noting that $2/(k-1) \leq 2$ when $k \geq 2$. ◀

▶ **Corollary 1.2.** *There exists a deterministic algorithm for learning-augmented online caching that achieves a competitive ratio of*

$$2\min\left(\min\left(1 + 2\frac{\eta}{\mathsf{OPT}}, 4 + \frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}\right), k\right).$$

Proof. Combine BlindOracle with LRU using the "combiner" from Theorem 2.1, with the performance of BlindOracle being bounded by Theorem 1.1. ◀

▶ **Corollary 1.3.** *There exists a randomized algorithm for learning-augmented online caching that achieves a competitive ratio of*

$$(1+\gamma)\min\left(\min\left(1 + 2\frac{\eta}{\mathsf{OPT}}, 4 + \frac{4}{k-1}\frac{\eta}{\mathsf{OPT}}\right), H_k\right)$$

for any $\gamma \in (0, 1/4)$.[7] *(Here, $H_k = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{k} = \ln(k) + O(1)$ is the k-th harmonic number.)*

Proof. Like in the proof above, combine BlindOracle with algorithm Equitable of Achlioptas et al. [1][8], this time using the "combiner" from Theorem 2.2. ◀

[7] The trade-off in γ and the cost is additive; thus, it does not factor into the competitive ratio.

[8] We use Equitable because it achieves the optimal worst-case competitive ratio of H_k for online caching; Marker has a competitive ratio of $2H_k - 1$ [1].

4 Lower Bound for Deterministic Algorithms

We now show that combining BlindOracle with LRU achieves an optimal competitive ratio bound (in terms of k and $\varepsilon = \eta/\mathsf{OPT}$) among all deterministic algorithms for learning-augmented online caching by proving Theorem 1.4:

▶ **Theorem 1.4.** *The competitive ratio bound for any deterministic learning-augmented online caching algorithm must be at least*

$$1 + \Omega\left(\min\left(\frac{1}{k}\frac{\eta}{\mathsf{OPT}}, k\right)\right)$$

in terms of k and η/OPT.

Proof. Fix a $k > 2$, and let $\mathcal{A}$ be any deterministic algorithm for learning-augmented online caching. We show that for any even integer $2j$ such that $1 < 2j < k$, there exists a sequence of inputs with $\mathsf{OPT}(\sigma)$ growing arbitrarily large that satisfies $\varepsilon/k = 2j$ and

$$\frac{\mathsf{ALG}_{\mathcal{A}}(\sigma, h)}{\mathsf{OPT}(\sigma)} \geq 1 + \frac{\varepsilon}{4k}$$

for all (σ, h) in the sequence. As $2j$ nears k, this lower bound on competitive ratio approaches $1 + k/4$, so this lower bound binds up to a competitive ratio of $\Omega(k)$, thus yielding the stated result.

We spend the remainder of this proof constructing such inputs (σ, h). Let $P_1, \ldots, P_k, Q$ be $k + 1$ distinct pages. We make the following sequence of requests, which we call a *phase*:

1. For $i = 1, \ldots, k - 1$:
   a. Request pages $P_1, \ldots, P_k$ in order, predicting each page to next appear k requests from now.
2. Request pages $P_1, \ldots, P_k$ in order, predicting each page to next appear $k + j + 2$ requests from now.
3. Request page Q and predict that it will next appear $j + 1$ requests from now.
4. For $i = 1, \ldots, j$:
   a. Request the page evicted by $\mathcal{A}$ during the previous request if it exists. Otherwise, request an arbitrary page. For each page, provide the same prediction as the last time this page was requested.
5. Request page Q and predict that it will next appear $k^2 + 1 + z$ requests from now, for some $z \geq 0$ that we will specify below.

As an outer loop, we can repeat the above "phase" as many times as needed.

In a single phase, observe that OPT makes at most two evictions: One to evict Q if it is in the cache at the beginning of the phase and one upon the arrival of Q in step (3). For the latter, it suffices to evict a page that does not appear among the at most $j + 1 \leq k$ pages requested in steps (4) and (5). Note also that OPT makes at least one eviction in any phase, since each phase involves serving $k + 1$ distinct pages.

On the other hand, I claim $\mathcal{A}$ makes at least $j + 1$ evictions. First, if the cache of $\mathcal{A}$ at the end of step (2) does not consist of pages $P_1, \ldots, P_k$, then $\mathcal{A}$ must have incurred cost at least $k \geq j + 1$ during steps (1) and (2). Thus, we may assume that $\mathcal{A}$'s cache consists exactly of pages $P_1, \ldots, P_k$ at the end of step (2). If so, $\mathcal{A}$ has to evict a page for each of the $j + 1$ requests made in steps (3) and (4).

Finally, observe that all the predictions are accurate except those made for the pages that arrive in step (4) and the prediction in step (5). I claim that each prediction for a page arriving in step (4) is off by at most $k + j < 2k$. Indeed, a request to Q will be off by

at most j, as Q is requested again in step (5). And a request to P_ℓ will be off by at most $k + j$, as either page P_ℓ appears in step (1) of the next phase or the sequence of requests terminates. To bound the error for the former, note that P_ℓ will appear at most k requests into step (1) of the next phase, which starts j requests after the first request of step (4). For the prediction made in step (5), note that it is off from the ground truth by exactly z.

Putting everything together, we have that $1 \leq \mathsf{OPT} \leq 2$ and $\eta \leq j \cdot 2k + z$. In fact, we can set z so that $\eta = 2jk \cdot \mathsf{OPT}$ exactly. In this case,

$$\mathsf{ALG}_{\mathcal{A}} \geq j + 1 = 2 + \frac{1}{2}\left(1 - \frac{1}{j}\right) \cdot 2j \geq \mathsf{OPT} + \frac{1}{4}\frac{\eta}{k},$$

with $\varepsilon/k = 2j$. To finish, notice that we can make OPT arbitrarily large by repeating the "phase" defined above multiple times in sequence; the same analysis holds. ◀

5 Conclusion

In this paper, we show that the simple approach of combining BlindOracle with competitive algorithms for online caching surprisingly achieves state-of-the-art performance for learning-augmented online caching. We additionally show that combining BlindOracle with LRU is optimal among deterministic algorithms for learning-augmented online caching. An open question is whether the bounds we achieve can be improved using randomization to match the lower bound of Rohatgi [25].

References

1. Dimitris Achlioptas, Marek Chrobak, and John Noga. Competitive analysis of randomized paging algorithms. *Theor. Comput. Sci.*, 234(1-2):203–218, 2000.
2. Antonios Antoniadis, Christian Coester, Marek Eliás, Adam Polak, and Bertrand Simon. Online metric algorithms with untrusted predictions. *CoRR*, abs/2003.02144, 2020. To appear at ICML 2020. `arXiv:2003.02144`.
3. Nikhil Bansal, Niv Buchbinder, and Joseph Naor. A primal-dual randomized algorithm for weighted paging. *J. ACM*, 59(4):19:1–19:24, 2012.
4. Nikhil Bansal, Niv Buchbinder, and Joseph Naor. Randomized competitive algorithms for generalized caching. *SIAM J. Comput.*, 41(2):391–414, 2012.
5. Avrim Blum and Carl Burch. On-line learning and the metrical task system problem. *Mach. Learn.*, 39(1):35–58, 2000.
6. Allan Borodin and Ran El-Yaniv. *Online computation and competitive analysis*. Cambridge University Press, 1998.
7. Joan Boyar, Lene M. Favrholdt, Christian Kudahl, Kim S. Larsen, and Jesper W. Mikkelsen. Online algorithms with advice: A survey. *ACM Comput. Surv.*, 50(2):19:1–19:34, 2017.
8. Sébastien Bubeck and Aleksandrs Slivkins. The best of both worlds: Stochastic and adversarial bandits. In *COLT 2012 - The 25th Annual Conference on Learning Theory, June 25-27, 2012, Edinburgh, Scotland*, pages 42.1–42.23, 2012.
9. Yihe Dong, Piotr Indyk, Ilya P. Razenshteyn, and Tal Wagner. Learning space partitions for nearest neighbor search. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.
10. 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.
11. Sreenivas Gollapudi and Debmalya Panigrahi. Online algorithms for rent-or-buy with expert advice. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 2319–2327, 2019.

12 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*, 2019.

13 Elias Koutsoupias and Christos H. Papadimitriou. Beyond competitive analysis. *SIAM J. Comput.*, 30(1):300–317, 2000.

14 Tim Kraska, Alex Beutel, Ed H. Chi, Jeffrey Dean, and Neoklis Polyzotis. The case for learned index structures. In *Proceedings of the 2018 International Conference on Management of Data, SIGMOD Conference 2018, Houston, TX, USA, June 10-15, 2018*, pages 489–504, 2018.

15 Ravi Kumar, Manish Purohit, Aaron Schild, Zoya Svitkina, and Erik Vee. Semi-online bipartite matching. In *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, pages 50:1–50:20, 2019.

16 Ravi Kumar, Manish Purohit, Zoya Svitkina, and Erik Vee. Interleaved caching with access graphs. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1846–1858, 2020.

17 Silvio Lattanzi, Thomas Lavastida, Benjamin Moseley, and Sergei Vassilvitskii. Online scheduling via learned weights. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1859–1877, 2020.

18 Thodoris Lykouris and Sergei Vassilvitskii. Competitive caching with machine learned advice. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, pages 3302–3311, 2018.

19 Mohammad Mahdian, Hamid Nazerzadeh, and Amin Saberi. Online optimization with uncertain information. *ACM Trans. Algorithms*, 8(1):2:1–2:29, 2012.

20 Andres Muñoz Medina and Sergei Vassilvitskii. Revenue optimization with approximate bid predictions. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 1858–1866, 2017.

21 Vahab S. Mirrokni, Shayan Oveis Gharan, and Morteza Zadimoghaddam. Simultaneous approximations for adversarial and stochastic online budgeted allocation. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 1690–1701, 2012.

22 Michael Mitzenmacher. A model for learned Bloom filters and optimizing by sandwiching. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 462–471, 2018.

23 Michael Mitzenmacher. Scheduling with predictions and the price of misprediction. In *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, pages 14:1–14:18, 2020.

24 Manish Purohit, Zoya Svitkina, and Ravi Kumar. Improving online algorithms via ML predictions. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, 3-8 December 2018, Montréal, Canada*, pages 9684–9693, 2018.

25 Dhruv Rohatgi. Near-optimal bounds for online caching with machine learned advice. In *Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, Salt Lake City, UT, USA, January 5-8, 2020*, pages 1834–1845, 2020.

26 Tim Roughgarden. Beyond worst-case analysis. *Commun. ACM*, 62(3):88–96, 2019.

27 Daniel Dominic Sleator and Robert Endre Tarjan. Amortized efficiency of list update and paging rules. *Commun. ACM*, 28(2):202–208, 1985.

A 4/3-Approximation Algorithm for the Minimum 2-Edge Connected Multisubgraph Problem in the Half-Integral Case

Sylvia Boyd
School of Electrical Engineering and Computer Science, University of Ottawa, Canada
sboyd@uottawa.ca

Joseph Cheriyan
Department of Combinatorics and Optimization, University of Waterloo, Canada
jcheriyan@uwaterloo.ca

Robert Cummings
Department of Combinatorics and Optimization, University of Waterloo, Canada

Logan Grout
Department of Combinatorics and Optimization, University of Waterloo, Canada

Sharat Ibrahimpur
Department of Combinatorics and Optimization, University of Waterloo, Canada
sharat.ibrahimpur@uwaterloo.ca

Zoltán Szigeti
University Grenoble Alpes, CNRS, G-SCOP, France
zoltan.szigeti@grenoble-inp.fr

Lu Wang
Department of Combinatorics and Optimization, University of Waterloo, Canada

Abstract

Given a connected undirected graph $\overline{G}$ on n vertices, and non-negative edge costs c, the 2ECM problem is that of finding a 2-edge connected spanning multisubgraph of $\overline{G}$ of minimum cost. The natural linear program (LP) for 2ECM, which coincides with the subtour LP for the Traveling Salesman Problem on the metric closure of $\overline{G}$, gives a lower bound on the optimal cost. For instances where this LP is optimized by a half-integral solution x, Carr and Ravi (1998) showed that the integrality gap is at most $\frac{4}{3}$: they show that the vector $\frac{4}{3}x$ dominates a convex combination of incidence vectors of 2-edge connected spanning multisubgraphs of $\overline{G}$.

We present a simpler proof of the result due to Carr and Ravi by applying an extension of Lovász's splitting-off theorem. Our proof naturally leads to a $\frac{4}{3}$-approximation algorithm for half-integral instances. Given a half-integral solution x to the LP for 2ECM, we give an $O(n^2)$-time algorithm to obtain a 2-edge connected spanning multisubgraph of $\overline{G}$ whose cost is at most $\frac{4}{3}c^T x$.

2012 ACM Subject Classification Mathematics of computing → Paths and connectivity problems; Mathematics of computing → Approximation algorithms; Theory of computation → Routing and network design problems

Keywords and phrases 2-Edge Connectivity, Approximation Algorithms, Subtour LP for TSP

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.61

Category APPROX

Funding *Joseph Cheriyan*: Supported by NSERC RGPIN-2019-04197.
Sharat Ibrahimpur: Supported in part by NSERC grant 327620-09.

Acknowledgements SB, JC, and SI thank BIRS, Canada for organizing the workshop on *The Traveling Salesman Problem* (2018), where some of the results in our references were presented.



© Sylvia Boyd, Joseph Cheriyan, Robert Cummings, Logan Grout, Sharat Ibrahimpur, Zoltán Szigeti, and Lu Wang;
licensed under Creative Commons License CC-BY



1 Introduction

The 2-edge connected multisubgraph (2ECM) problem is a fundamental problem in survivable network design where one wants to be resilient against a single edge failure. In this problem, we are given an undirected graph $\overline{G} = (\overline{V}, \overline{E})$ with non-negative edge costs c and we want to find a 2-edge connected spanning multisubgraph of $\overline{G}$ of minimum cost. Below we give an integer linear program for 2ECM. The variable x_e denotes the number of copies of edge e that are used in a feasible solution. For any $S \subset V$, $\delta(S) := \{e = uv \in \overline{E} : u \in S, v \notin S\}$ denotes the cut induced by S. For any $F \subseteq E$ and vector $x \in \mathbb{R}^E$, we use $x(F)$ as a shorthand for $\sum_{e \in F} x_e$. Also, for any graph H with edge costs c, we sometimes use $c(H)$ as a shorthand for $c(E(H))$.

$$\text{(2ECM-IP)} \qquad \begin{aligned} \min \quad & \sum_{e \in \overline{E}} c_e x_e && && (1)\\ \text{subject to} \quad & x(\delta(S)) \geq 2 && \forall \emptyset \subsetneq S \subsetneq \overline{V}, && (2)\\ & x_e \geq 0 && \forall e \in \overline{E}, && (3)\\ & x_e \text{ integral} && \forall e \in \overline{E}. && (4) \end{aligned}$$

It is easy to see that an optimal solution for 2ECM never has to use more than two copies of an edge. As is discussed in [6], since we are allowed to use more than one copy of an edge, without loss of generality, we may assume that $\overline{G}$ is complete by performing the metric completion: for each $u, v \in \overline{V}$ we set the new cost of the edge uv to be the shortest path distance between u and v in $\overline{G}$. In the sequel, we assume that $\overline{G}$ is a complete graph and that the cost function c is metric i.e., $c \geq 0$ and for every $u, v, w \in \overline{V}$, we have $c_{uw} \leq c_{uv} + c_{vw}$.

The linear relaxation (2ECM-LP) for 2ECM is obtained by dropping the integrality constraints given by (4). By a result due to Goemans and Bertsimas [8] called the *parsimonious property*, adding the constraint $x(\delta(v)) = 2$ for each $v \in \overline{V}$ to (2ECM-LP) does not increase the optimal solution value; here, we require the assumption that the costs form a metric. So, the optimal value of (2ECM-LP) is the same as the optimal value for the well-known subtour elimination LP (Subtour-LP) for the Traveling Salesman Problem (TSP) defined below. Due to this connection, we often refer to an optimal solution for (2ECM-LP) as an optimal solution to (Subtour-LP), and vice versa. Another consequence of the parsimonious property is that for graphs with at least 3 vertices, the constraint $x_e \leq 1$ is implied by (Subtour-LP): for any $e = uv$, we have $2x_e = x(\delta(u)) + x(\delta(v)) - x(\delta(\{u, v\})) \leq 2$.

$$\text{(Subtour-LP)} \qquad \begin{aligned} \min \quad & \sum_{e \in \overline{E}} c_e x_e && && (5)\\ \text{subject to} \quad & x(\delta(S)) \geq 2 && \forall \emptyset \subsetneq S \subsetneq \overline{V}, && (6)\\ & x(\delta(v)) = 2 && \forall v \in \overline{V}, && (7)\\ & x_e \geq 0 && \forall e \in \overline{E}. && (8) \end{aligned}$$

A long-standing open problem called the "four-thirds conjecture" states that the integrality gap of (Subtour-LP) is $\frac{4}{3}$. Besides the importance of 2ECM in the field of survivable network design, the connection between (2ECM-LP) and (Subtour-LP) has spurred interest in determining the integrality gap for (2ECM-LP) as a means to gaining useful lower bounds on the integrality gap for (Subtour-LP). The general version of metric TSP has resisted all attempts at proving an upper bound better than $\frac{3}{2}$ on the integrality gap, so a great deal of research has focused on obtaining improvements for important special cases. In [13], the authors conjecture that the integrality gap for (Subtour-LP) is achieved on instances where

an optimal (fractional) solution to (Subtour-LP) is half integral i.e., $2x_e \in \mathbb{Z}_{\geq 0}$ for all e. We refer to such instances as *half integral instances*. More than two decades ago, Carr and Ravi [6] proved that the integrality gap of (2ECM-LP) is at most $\frac{4}{3}$ in the half-integral case. They show that $\frac{4}{3}x$ dominates a convex combination of 2-edge connected spanning multisubgraphs of $\overline{G}$. This supports the four-thirds conjecture for TSP since the (integer) optimal value for 2ECM lower bounds the (integer) optimal value for TSP. However, the proof of Carr and Ravi does not give a polynomial-time algorithm for 2ECM. Very recently, in [12], Karlin, Klein, and Oveis Gharan gave a randomized approximation algorithm for half-integral instances of TSP whose (expected) approximation factor is $\frac{3}{2} - 0.00007$. This immediately implies a better than $\frac{3}{2}$-approximation algorithm, albeit randomized, for 2ECM as well.

We note that the result of Carr and Ravi mentioned above does not apply to the strict variant of 2ECM (henceforth denoted by 2ECS) where we are allowed to pick at most one copy of an edge in $\overline{G}$, i.e. where we are considering subgraphs of $\overline{G}$ rather than multisubgraphs; similarly, our main result does not apply to 2ECS.

1.1 Our Work

Our main contribution is a deterministic approximation algorithm for 2ECM on half-integral instances that matches the existence result in [6].

▶ **Theorem 1.** *Let x denote an optimal half-integral solution to an instance $(\overline{G}, c)$ of* (Subtour-LP) *(and* (2ECM-LP)*). There is an $O(|V(\overline{G})|^2)$-time algorithm for computing a 2-edge connected spanning multisubgraph of $\overline{G}$ with cost at most $\frac{4}{3}c^T x$.*

Given a half-integral solution x to (Subtour-LP) for $\overline{G}$, let $G = (V, E)$ denote the multigraph induced by $2x$. Formally, the vertex-set $V := \overline{V}$, and for each edge $e \in \overline{E}$, the edge-set E has $2x_e$ copies of the edge e. Note that if $|\overline{V}| \geq 3$, then $2x_e \in \{0, 1, 2\}$ for all $e \in \overline{E}$, and if $|\overline{V}| = 2$, then $2x_e = 4$ for the unique edge $e \in \overline{E}$. With a slight abuse of notation, we use the same cost function c to denote the edge costs in G i.e., $c_f := c_e$ where $e \in \overline{E}$ gave rise to the edge $f \in E$. By (7) and (6), G is a 4-regular 4-edge connected multigraph. Theorem 1 follows from the following result applied to the graph G induced by $2x$.

▶ **Theorem 2.** *Let $G = (V, E)$ be a 4-regular 4-edge connected multigraph on n vertices. Let $c : E \to \mathbb{R}$ be an arbitrary cost function on the edges of G (negative costs on the edges are allowed), and let e be an arbitrary edge in G. Then, in $O(n^2)$ time, we can find a 2-edge connected spanning subgraph H of $G - e$ satisfying:*

- **(i)** *$c(H) \leq \frac{2}{3}c(G - e)$; and*
- **(ii)** *each multiedge of G appears at most once in H (multiedges may arise in H due to multiedges in G).*

For any $F \subseteq E$, let $\chi^F \in \{0, 1\}^E$ denote the characteristic vector of F: $\chi^F_e = 1$ if and only if $e \in F$. Note that distinct multiedges in E correspond to distinct coordinates in χ^F. As mentioned before, Carr and Ravi prove the existence of such a subgraph H by showing that for any 4-regular 4-edge connected multigraph G, there exists a finite collection $H_1, \ldots, H_k$ of 2-edge connected spanning subgraphs of G such that $\frac{2}{3}\chi^{E(G)\setminus\{e\}}$ lies in the convex hull of $\{\chi^{H_i}\}_i$. At a high level, their proof is inductive and splits into two cases based on whether G has a certain kind of a tight set (a cut of size 4). In the first case they construct two smaller instances of the problem by contracting each of the shores of the tight set, and in the second case they perform two distinct splitting-off operations at a designated vertex to obtain two smaller instances of the problem. In either case, the convex combinations from the

two subinstances are merged to obtain a convex combination for G. The first case requires gluing since the subgraphs obtained from the two subinstances need to agree on a (tight) cut. Merging the convex combinations arising from the second case is rather straightforward as the two subinstances are more or less independent.

Our first insight in this work is that the case from Carr and Ravi's proof that requires the gluing step can be completely avoided, thereby unifying the analysis. This is discussed in Section 2. Our proof relies on an extension of Lovász's splitting-off theorem that is due to Bang-Jensen et al., [2]. For further discussion on splitting-off theorems, see [7, Chapter 8]. The challenge in efficiently finding a cheap subgraph H from the above convex combination construction is that each inductive step requires solving two subinstances of the problem, each with one fewer vertex, leading to an exponential-time algorithm. Having said that, an (expected) polynomial-time Las Vegas randomized algorithm can be easily designed that randomly recurses on one of the two subinstances and produces a 2-edge connected spanning subgraph whose expected cost is at most $\frac{2}{3}c(G-e)$. Our second insight, which is used in derandomizing the above procedure, is that it is easy to recognize which of the two subinstances leads to a "cheaper" solution, so we recurse only on the cheaper subinstance. Complementing this step, we lift the solution back to the original instance. This operation can lead to two different outcomes so the cost analysis must account for the worst outcome. There is a choice of defining the costs in the subinstance such that the cost of the lifted subgraph is the same irrespective of the outcome. Such a choice can lead to negative costs, but this is not a hindrance for our inductive step because Theorem 2 allows arbitrary real-valued edge costs. This generality of cost functions is crucial to our algorithm.

In Section 4 we consider a well-studied special case of the 2ECM problem. We present a simple $O(n^3)$-time algorithm that given a 3-regular 3-edge connected graph G, finds a 2-edge connected spanning multisubgraph of cost at most $\frac{7}{8}c(G)$ (see Theorem 10). The proof is inspired by that of Haddadan, Newman, and Ravi in [11] where they give a polynomial-time algorithm for this problem with a factor $\frac{15}{17}(> \frac{7}{8})$. In [10, Theorem 1.1], Haddadan and Newman improve this result to a factor $\frac{7}{8}$, and very recently, in [9, Theorem 1.20], Haddadan claims a stronger result with a factor of $\frac{41}{47} = \frac{7}{8} - \frac{1}{376}$. We remark that these proofs are longer and/or more complicated than that of Theorem 10. Another motivation for Section 4 is to illustrate the potential of Theorem 2 in giving simpler proofs for results that may not have any explicit half-integrality restrictions.

1.2 Related Work

The 2ECM problem has been intensively studied in network design and several works have tried to bound the integrality gap α2ECM of (2ECM-LP). For the general case with metric costs, we have $\frac{6}{5} \leq \alpha\text{2ECM} \leq \frac{3}{2}$, where the lower bound is from [1] and the upper bound follows from the polyhedral analysis of Wolsey [16] and Shmoys and Williamson [15] (this analysis also gives a $\frac{3}{2}$-approximation algorithm). It is generally conjectured that $\alpha\text{2ECM} = \frac{4}{3}$, however in [1], Alexander et al., study α2ECM and conjecture that $\alpha\text{2ECM} = \frac{6}{5}$ based on their findings. As mentioned before, Carr and Ravi [6] show that the integrality gap of (2ECM-LP) is at most $\frac{4}{3}$ in the half-integral case. In [5] Boyd and Legault consider a more restrictive collection of instances called half-triangle instances where the optimal LP solution is half-integral and the graph induced by the half-edges is a collection of disjoint triangles. They prove that $\alpha\text{2ECM} = \frac{6}{5}$ in this setting. Half-triangle solutions are of interest as there is evidence that the integrality gap of (2ECM-LP) is attained at such solutions (see [1]). When the costs come from a graphic metric (i.e., we want to find a minimum-size 2-edge connected spanning multisubgraph of a given unweighted graph), we have $\frac{8}{7} \leq \alpha\text{2ECM} \leq \frac{4}{3}$ (see [3, 14]).

2 A Simpler Proof of a Result of Carr and Ravi

In this section, we give a simplified proof of the following result from [6]. As mentioned before, avoiding the case involving the gluing operation is useful for our algorithm in Section 3. For notational convenience, for any subgraph K of some graph, we use χ^K to denote $\chi^{E(K)}$ whenever the underlying graph is clear from the context.

▶ **Theorem 3** (Statement 1 from [6]). *Let $G = (V, E)$ be a 4-regular 4-edge connected multigraph and $e = uv$ be an arbitrary edge in this graph. There exists a finite collection $\{H_1, \ldots, H_k\}$ of 2-edge connected spanning subgraphs of $G - e$ such that for some nonnegative $\mu_1, \ldots, \mu_k$ with $\sum_i \mu_i = 1$, we have $\frac{2}{3}\chi^{E\setminus\{e\}} = \sum_{i=1}^{k} \mu_i \chi^{H_i}$. Moreover, we may assume that none of the H_i's use more than one copy of an edge in E; H_i may have multiedges as long as they come from distinct edges in G.*

2.1 Operations involving splitting-off at a vertex

The following tools on the splitting-off operation will be useful. In keeping with standard terminology, we designate a vertex v (one of the endpoints of e in the theorem statement) at which the splitting-off operation is applied. For a multigraph $H = (V, E)$ and $x, y \in V$, let $\lambda_H(x, y)$ denote the size of a minimum (x, y)-cut in H, and let $\deg_H(x)$ denote the degree of x in H. Note that each multiedge is counted separately towards the degree of a vertex and the size of a cut.

▶ **Definition 4.** *Given a multigraph G and two edges sv and vt that share an endpoint v, the graph $G_{s,t}$ obtained by* splitting off *the pair (sv, vt) at v is given by $G + st - sv - vt$.*

▶ **Definition 5.** *Given a multigraph G and a vertex v of G of even degree, a* complete splitting at *v is a sequence of $\frac{1}{2}deg_G(v)$ splitting off operations that result in vertex v having degree zero in the resulting graph.*

▶ **Definition 6.** *Let $k \geq 2$ be an integer and let G be a multigraph such that for all $x, y \in V \setminus \{v\}$, $\lambda_G(x, y) \geq k$. Let $e = sv$ and vt be two edges incident to v. We say that the pair (sv, vt) is* admissible *if for all $x, y \in V \setminus \{v\}$, $\lambda_{G_{s,t}}(x, y) \geq k$, and for a particular edge $e \in \delta(v)$, we let A_e denote the set of edges $f \in \delta(v) \setminus \{e\}$ such that (e, f) is an admissible pair.*

The following result due to Bang-Jensen et al., [2] shows that in our setting with a 4-regular 4-edge connected multigraph at least two distinct edges incident to v form an admissible pair with $e = uv$. Using this we can perform a complete splitting at v in two distinct ways.

▶ **Lemma 7** (Theorem 2.12 from [2]). *Let $k \geq 2$ be an even integer. Let G be a multigraph such that for all $x, y \in V \setminus \{v\}$, $\lambda_G(x, y) \geq k$. Let $\deg_G(v)$ be even (each multiedge is counted separately towards the degree). Then, $|A_{uv}| \geq \frac{1}{2}\deg_G(v)$.*

▶ **Lemma 8.** *Let G be a 4-regular 4-edge connected multigraph and $e = vx$ be an edge incident to v. Then, (i) $|A_e| \geq 2$; and (ii) if (e, f) is an admissible pair for some $f = vy \in \delta(v) \setminus \{e\}$, then the remaining two edges in $\delta(v) \setminus \{e, f\}$ form an admissible pair in $G_{x,y}$.*

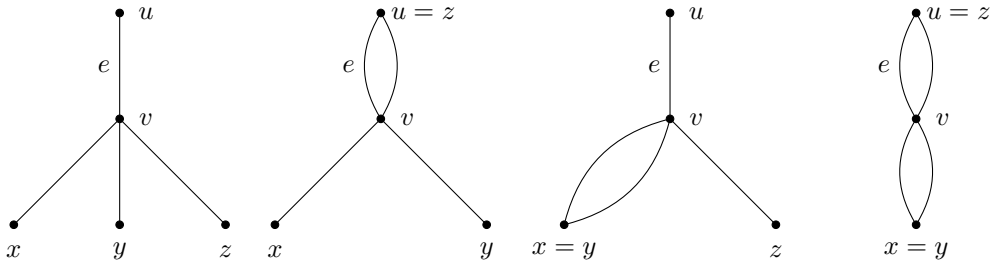
Proof. Conclusion (i) follows from Lemma 7 since G is 4-regular and 4-edge connected. For conclusion (ii), let $f \in \delta(v) \setminus \{e\}$ be such that (e, f) forms an admissible pair in G. Let $G_{x,y}$ denote the graph obtained by splitting off the pair $(e = vx, f = vy)$ i.e.,

$G_{x,y} = G - vx - vy + xy$. Observe that the hypothesis of Lemma 7 still holds for $G_{x,y}$ with $k = 4$ because (a) we performed a splitting off operation using an admissible pair of edges; and (b) $\deg_{G_{x,y}}(v) = 2$ is even. Let g denote one of the two remaining edges in $\delta(v) \setminus \{e, f\}$. By Lemma 7, the other unique edge $h \in \delta(v) \setminus \{e, f, g\}$ forms an admissible pair with g in $G_{x,y}$. ◀

Equipped with the above tools, we give a proof of Theorem 3.

Proof of Theorem 3. Let $G = (V, E)$ be a 4-regular 4-edge connected multigraph and let $e = uv$ be an arbitrary edge in G. We prove this theorem via induction on $n := |V(G)|$. The base case $n = 2$ corresponds to a pair of vertices having four parallel edges, call them e, f, g, h. Observe that $\frac{2}{3}\chi^{E\setminus\{e\}} = \frac{1}{3}\big(\chi^{\{f,g\}} + \chi^{\{f,h\}} + \chi^{\{g,h\}}\big)$, so the induction hypothesis is true for the base case.

For the induction step, suppose that $n \geq 3$ and the hypothesis holds for all 4-regular 4-edge connected multigraphs with at most $n - 1$ vertices and for all choices of $e \in E$. Consider a 4-regular 4-edge connected multigraph G on n vertices and an arbitrary edge $e = uv \in E(G)$. Besides e, let vx, vy, vz be the other three edges incident to v. With a relabeling of vertices, by Lemma 8, we may assume that (uv, vx) and (uv, vy) form an admissible pair in G (see Figure 1).



(a) v has four distinct neighbors $|\mathsf{A_e}| \in \{2, 3\}$.

(b) v has two parallel edges with u $\mathsf{A_e} = \{vx, vy\}$.

(c) v has two parallel edges with $x, x \neq u$ $\mathsf{A_e} = \{vx, vy\}$.

(d) v has two parallel edges to each of $\{u, x\}$ $\mathsf{A_e} = \{vx, vy\}$.

Figure 1 Four configurations of edges in $\delta(v) = \{uv, vx, vy, vz\}$ that can arise in our proof.

By the second conclusion of Lemma 8, (vy, vz) is an admissible pair in $G_{u,x}$, and (vx, vz) is an admissible pair in $G_{u,y}$. Consider the graph G_1 obtained by splitting off the pair (vy, vz) in $G_{u,x}$ i.e., $G_1 = G - v + \{ux, yz\}$; it is customary to drop the vertex v after all its edges have been split off. Similarly, let G_2 be the graph obtained by splitting off the pair (vx, vz) in $G_{u,y}$ i.e., $G_2 = G - v + \{uy, xz\}$.

Since we only split off admissible pairs, both G_1 and G_2 are 4-regular 4-edge connected multigraphs on $n - 1$ vertices. Recall that for any subgraph K of some graph, χ^K is a shorthand for $\chi^{E(K)}$ whenever the underlying graph is clear from the context. Applying the induction hypothesis to G_1 with the designated edge $e_1 = ux$ gives:

$$\frac{2}{3} \cdot \chi^{E(G_1)\setminus\{e_1\}} = \frac{2}{3} \cdot \chi^{(E\setminus\delta(v))\cup\{yz\}} = \sum_{i=1}^{k_1} \mu_i^1 \chi^{H_i^1} \,, \qquad \text{(ConvexComb-}G_1\text{)}$$

where $\{\mu_i^1\}_i$ denote the coefficients in a convex combination, and $\{H_i^1\}_i$ are 2-edge connected spanning subgraphs of G_1 such that none of them use more than one copy of an edge in G_1.

Repeating the same argument for G_2 with the designated edge $e_2 = uy$ gives:

$$\frac{2}{3} \cdot \chi^{E(G_2)\setminus\{e_2\}} = \frac{2}{3} \cdot \chi^{(E\setminus\delta(v))\cup\{xz\}} = \sum_{i=1}^{k_2} \mu_i^2 \chi^{H_i^2} \ , \tag{ConvexComb-G_2}$$

where $\{\mu_i^2\}_i$ denote the coefficients in the other convex combination arising from $\{H_i^2\}_i$. It remains to combine (ConvexComb-G_1) and (ConvexComb-G_2) to obtain such a representation for G with the designated edge e. We mimic the strategy from [6].

For each $i \in \{1, \ldots, k_1\}$, we lift H_i^1 to a spanning subgraph $\hat{H}_i^1$ of $G - e$. Define $\hat{H}_i^1$ as follows:

$$\hat{H}_i^1 := \begin{cases} H_i^1 - yz + vy + vz & \text{if } yz \in E(H_i^1), \\ H_i^1 + vy + vx & \text{if } yz \notin E(H_i^1). \end{cases} \tag{Lift-G_1}$$

Similarly, for each $i \in \{1, \ldots, k_2\}$, we define $\hat{H}_i^2$ as the following spanning subgraph of $G - e$:

$$\hat{H}_i^2 := \begin{cases} H_i^2 - xz + vx + vz & \text{if } xz \in E(H_i^2), \\ H_i^2 + vx + vy & \text{if } xz \notin E(H_i^2). \end{cases} \tag{Lift-G_2}$$

We finish the proof of Theorem 3 by arguing that the following convex combination meets all the requirements:

$$q := \frac{1}{2} \sum_{i=1}^{k_1} \mu_i^1 \chi^{\hat{H}_i^1} + \frac{1}{2} \sum_{i=1}^{k_2} \mu_i^2 \chi^{\hat{H}_i^2} \ . \tag{ConvexComb-G}$$

Many of our arguments are the same for G_1 and G_2 so we just mention them in the context of G_1. First of all, by the induction hypothesis and (Lift-G_1) it is clear that $e(= uv), yz, ux \notin E(\hat{H}_i^1)$, where yz and ux refer to the edges that originated from the splitting off operations applied at v. Next, we argue that $\hat{H}_i^1$ is a spanning subgraph of G that uses no more than one copy of any edge in G. By the induction hypothesis, none of the subgraphs H_i^1 use more than one copy of an edge in G_1, and H_i^1 spans $V \setminus \{v\}$. By the way we lift H_i^1 to $\hat{H}_i^1$, it is clear that $\hat{H}_i^1$ uses no more than one copy of any multiedge in G, and that it is spanning. To see that $\hat{H}_i^1$ is 2-edge connected, observe that the two cases of lifting may be viewed as either (i) subdividing the edge yz by a node v when $yz \in E(H_i^1)$, or (ii) adding an edge yx and subdividing it by a node v when $yz \notin E(H_i^1)$. Clearly, these operations preserve 2-edge connectivity, hence, $\hat{H}_i^1$ is 2-edge connected.

It remains to argue that the vector q in the expression (ConvexComb-G) matches the vector $\frac{2}{3}\chi^{E(G)\setminus\{e\}}$. Since $\{\mu_i^1\}_i$ and $\{\mu_i^2\}_i$ denote coefficients in a convex combination, taking an unweighted average of these two combinations gives us another convex combination. Since none of the edges in $E(G) \setminus \delta(v)$ are modified in the lifting step, $q_f = 2/3$ for any such edge f. Next, consider the edge vy. Observe that $\hat{H}_i^1$ always contains the edge vy, whereas $\hat{H}_i^2$ contains vy only when $xz \notin E(H_i^2)$ (this happens with weight $1/3$). Therefore, $q_{vy} = \frac{1}{2} \cdot 1 + \frac{1}{2} \cdot \frac{1}{3} = \frac{2}{3}$. The analysis for vx is symmetric. Lastly, consider the edge vz. It appears in $\hat{H}_i^1$ ($\hat{H}_i^2$) if and only if $yz \in E(H_i^1)$ (respectively, $xz \in E(H_i^2)$). Therefore, $q_{vz} = \frac{1}{2} \cdot \frac{2}{3} + \frac{1}{2} \cdot \frac{2}{3} = \frac{2}{3}$. This completes the proof of Theorem 3. ◀

3 Our Algorithm and the Proof of Theorem 2

In this section we give a proof of Theorem 2 which we use to obtain a $\frac{4}{3}$-approximation algorithm for 2ECM on half-integral instances (Theorem 1). We apply the same splitting-off theorem of [2] together with an induction scheme that is captured in Theorem 2. A key feature of this theorem is that we allow edges of negative cost, although the edge costs in any instance of 2ECM are non-negative.

Consider a 4-regular 4-edge connected multigraph $G = (V, E)$ on n vertices, and let $e = uv$ be an edge in G. Let $c : E \to \mathbb{R}$ be an arbitrary real-valued cost function. Our goal is to obtain a 2-edge connected spanning subgraph H of G whose cost is at most $\frac{2}{3}c(G - e)$ while ensuring that H uses no more than one copy of any multiedge in G. Observe that if we had access to the collection $\{H_1, \ldots, H_k\}$ of 2-edge connected spanning subgraphs from Theorem 3 for some k that is polynomial in $|V(G)|$, then we would be done: for any cost function c, the cheapest subgraph in this collection (w.r.t. cost c) is one such desired subgraph. It is not clear how to efficiently obtain such a collection; a naive algorithm that follows the proof of Theorem 3 does not run in polynomial time.

As alluded to before, for the purposes of obtaining a cheap 2-edge connected subgraph, it suffices to only recurse on one of two subinstances that arise in the proof of Theorem 3. This insight comes from working backwards from (ConvexComb-G). Since this convex combination for G is a simple average of convex combinations from two subinstances (see (ConvexComb-G_1) and (ConvexComb-G_2)), it is judicious to only recurse on the "cheaper" subinstance. Combining (ConvexComb-G_1) and (Lift-G_1), we get that the first subinstance gives rise to a convex combination for $\frac{2}{3}\chi^{E(G)\setminus\{e\}} + \frac{1}{3}(\chi^{\{vy\}} - \chi^{\{vx\}})$. On the other hand, the second subinstance gives rise to a convex combination for $\frac{2}{3}\chi^{E(G)\setminus\{e\}} + \frac{1}{3}(\chi^{\{vx\}} - \chi^{\{vy\}})$. Thus, we should recurse on G_1 if $c_{vx} \geq c_{vy}$, and G_2 otherwise. For the sake of argument, suppose that we are recursing on G_1. So far, we have ignored an important detail in the recursion: the splitting-off operation creates a new edge yz that was not originally present in G, so we need to assign it some cost to apply the algorithm recursively. Depending on how we choose the cost of yz, it might either be included or excluded from the subgraph obtained for the smaller instance, so to bound the cost of the lifted solution we must have a handle on both outcomes of the lift operation. Setting $c_{yz} := c_{vz} - c_{vx}$ balances the cost of both outcomes. Note that c_{yz} could possibly be negative, but this is permissible since the statement of Theorem 2 allows for arbitrary edge costs. We formalize the above ideas.

In the recursive step, we pick one end vertex v of e and apply a complete splitting off operation at v to obtain a 4-regular 4-edge connected graph on $n - 1$ vertices; this can be implemented in $O(n)$ time. The running time of the algorithm is $O(n^2)$, since we apply the induction step $O(n)$ times.

Let $T = \{v, x, y, z\}$ be the four neighbors of v and let $e = uv$. Recall that A_e denotes the set of edges $f \in \delta(v) \setminus \{e\}$ such that (e, f) is an admissible pair (see Definition 6).

▶ **Lemma 9.** *For $vx \in \delta(v) \setminus \{e\}$, we can check whether $vx \in \mathsf{A_e}$ in $O(n)$ time.*

Proof. We may suppose that the elements of the set T of neighbors of v are all distinct. Otherwise, by Lemma 8, we know exactly which pairs are admissible, see Figure 1. Consider the graph $\hat{G} = (G_{u,x})_{y,z}$ obtained by splitting off the pairs (uv, vx) and (yv, vz) at v. Let G^* be the graph obtained from $\hat{G}$ by contracting ux to a single vertex s and contracting yz to a single vertex t. Then we apply a max $s - t$ flow computation to check whether G^* has ≥ 4 edge-disjoint $s - t$ paths; otherwise, G^* has an (s, t)-cut $\delta(S)$ of size ≤ 3. In the latter case, it is clear that our trial splitting is not admissible.

In the former case, we claim that our trial splitting is admissible. Suppose that $\hat{G}$ is not 4-edge connected. Then there exists a non-empty, proper vertex set S in $\hat{G}$ such that $|T \cap S| \leq |T \setminus S|$ and $|\delta_{\hat{G}}(S)| < 4$. Clearly, $|S \cap T| \leq 2$, and if $|S \cap T| = 2$, then we have $|S \cap \{u, x\}| = 1$ and $|S \cap \{y, z\}| = 1$ (otherwise, S would give an (s,t)-cut of G^* of size ≤ 3). Since the size of the cut of S is the same in G and in $\hat{G}$, we have, by 4-edge connectivity of G, $4 > |\delta_{\hat{G}}(S)| = |\delta_G(S)| \geq 4$, a contradiction.

To see that the running time is linear, observe that G^* has $\leq 2n$ edges, an $s-t$ flow of value ≥ 4 can be computed by finding 4 augmenting paths, and each augmenting path can be found in linear time. ◀

Proof of Theorem 2. First, consider the base case in the recursion when $n = 2$. The only such 4-regular 4-edge connected multigraph is given by four parallel edges between u and v, of which e is one. Picking the two cheapest edges from the remaining three edges gives the desired subgraph.

For the induction step, suppose that $n \geq 3$ and the induction hypothesis holds for all 4-regular 4-edge connected multigraphs with at most $n-1$ vertices and for all choices of edge e. Consider a 4-regular 4-edge connected multigraph G on n vertices and an edge $e = uv$ in G.

Our algorithm proceeds as follows. By Lemmas 8 and 9, we can find in $O(n)$-time two neighbors of v, say x and y, such that $vx, vy \in A_e$ and $c_{vx} \geq c_{vy}$. Next, we construct the graph $\hat{G} := (G_{u,x})_{y,z} = G - v + \{ux, yz\}$ and extend the cost function c to the new edge yz as $c_{yz} := c_{vz} - c_{vx}$ (note that the cost of ux is inconsequential and that c_{yz} may be negative or non-negative). We recursively find a 2-edge connected spanning subgraph $\hat{H}$ of $\hat{G}$ with cost at most $\frac{2}{3}c(\hat{G} - ux)$. Then, we lift $\hat{H}$ to obtain a spanning subgraph H of G:

$$H := \begin{cases} \hat{H} - yz + vy + vz & \text{if } yz \in E(\hat{H}), \\ \hat{H} + vy + vx & \text{if } yz \notin E(\hat{H}). \end{cases}$$

We analyze the cost of this subgraph. Regardless of the cases above, our choice of c_{yz} implies that $c(H) = c(\hat{H}) + c_{vy} + c_{vx}$. Therefore,

$$\begin{aligned} c(H) \leq \frac{2}{3}c(\hat{G} - ux) + c_{vy} + c_{vx} &= \frac{2}{3}\{c(G-e) - c_{vx} - c_{vy} - c_{vz} + (c_{vz} - c_{vx})\} + c_{vy} + c_{vx} \\ &= \frac{2}{3}c(G-e) + \frac{1}{3}(c_{vy} - c_{vx}) \leq \frac{2}{3}c(G-e)\ , \end{aligned}$$

where the last inequality follows from our choice of vx, vy to satisfy $c_{vx} \geq c_{vy}$.

It remains to argue that H is a 2-edge connected spanning subgraph of $G - e$ that uses no more than one copy of any multiedge in G. It is clear that the following hold: (a) $e \notin E(H)$; (b) H is a spanning subgraph of G; and (c) each multiedge of G appears at most once in H. Since $\hat{H}$ is 2-edge connected and adding and/or subdividing an edge preserves 2-edge connectivity, H is 2-edge connected. Overall, in $O(n^2)$-time we have constructed a 2-edge connected spanning subgraph H of $G - e$ whose cost is at most $\frac{2}{3}c(G-e)$, thereby proving Theorem 2. ◀

Using Theorem 2, we give a deterministic $\frac{4}{3}$-approximation algorithm for 2ECM on half-integral instances.

Proof of Theorem 1. Let x be an optimal half-integral solution to (Subtour-LP) (and (2ECM-LP)) for an instance given by an n-vertex graph $\overline{G} = (\overline{V}, \overline{E})$ and a metric cost function c. Let $G = (V, E)$ denote the graph induced by $2x$ where for each $e \in E$ we include

$2x_e$ copies of the edge e in G. Since x has (fractional) degree 2 at each vertex and it is fractionally 2-edge connected, G is a 4-regular 4-edge connected multigraph. With a slight abuse of notation, we use the same cost function for the edges of E: for any $e \in E$, $c_e := c_f$, where f denotes the edge in $\overline{E}$ that gave rise to e. We invoke Theorem 2 on G and some edge $e \in E$. This gives us a 2-edge connected spanning subgraph H of $G - e$ satisfying $c(H) \leq \frac{2}{3}c(G-e)$. Lifting the subgraph H to $\overline{G}$ gives a 2-edge connected spanning multisubgraph $\overline{H}$ (of $\overline{G}$); note that $\overline{H}$ uses at most two copies of any edge in $\overline{G}$. By the first conclusion of Theorem 2 and the non-negativity of c, $c(\overline{H}) = c(H) \leq \frac{2}{3}c(G-e) \leq \frac{2}{3}c(G) = \frac{4}{3}c^T x$, where the last equality follows by recalling that G is induced by $2x$. Besides invoking Theorem 2 we only perform trivial graph operations so the running time is $O(n^2)$. ◀

4 2ECM for 3-Regular 3-Edge Connected Graphs

Let $G = (V, E)$ be a 3-regular 3-edge connected graph with non-negative edge costs $c \in \mathbb{R}^E_{\geq 0}$. In this section we consider an analogous problem to that of Theorem 2, namely the problem of finding a polynomial-time algorithm which gives a 2-edge connected spanning multisubgraph of G of cost at most $\beta c(G)$ for some $\beta \geq 0$. Note that the everywhere $\frac{2}{3}$ vector for G is feasible for (Subtour-LP). For any costs c for which the everywhere $\frac{2}{3}$ vector is also optimal for (Subtour-LP) (such as for the graphic metric), such an algorithm would provide a $\frac{3\beta}{2}$-approximation for 2ECM. The conjecture that α2ECM $= \frac{4}{3}$ would then imply $\beta = \frac{4}{3} \cdot \frac{2}{3} = \frac{8}{9}$ should be possible, and the $\frac{6}{5}$ conjecture for α2ECM would imply $\beta = \frac{6}{5} \cdot \frac{2}{3} = \frac{4}{5}$ should be possible. In [5] a constructive algorithm for $\beta = \frac{4}{5}$ is given, however it does not run in polynomial time.

In [11, Theorem 2], Haddadan, Newman, and Ravi show that it is possible to do better than $\frac{8}{9}$ for this problem, and provide an efficient algorithm for $\beta = \frac{15}{17}$. In fact, they show that the everywhere $\frac{15}{17}$ vector can be expressed as a convex combination of 2-edge connected spanning multisubgraphs of G and this convex combination can be found in polynomial time. They remark that combining their ideas with an efficient algorithm for Theorem 3 would imply the result for $\beta = \frac{7}{8} (< \frac{15}{17})$. Although a polynomial-time algorithm for Theorem 3 is not currently known, it is possible to use our result in Theorem 2 to obtain $\beta = \frac{7}{8}$, as follows.

▶ **Theorem 10.** *Let $G = (V, E)$ be a 3-regular 3-edge connected graph on n vertices with non-negative edge costs $c \in \mathbb{R}^E_{\geq 0}$. Then in $O(n^3)$-time we can find a 2-edge connected spanning multisubgraph H of G such that $c(H) \leq \frac{7}{8}c(G)$.*

Proof. Let F be a 2-factor of G that intersects all of the 3-edge cuts and 4-edge cuts of G. Such a 2-factor can be found in $O(n^3)$-time (see [4, Theorem 5.4]). Let G' be the graph obtained by contracting the cycles of F and removing any resulting loops, and let $M := E(G')$. Clearly G' is 5-edge connected (by choice of F), and thus the vector $y \in \mathbb{R}^M_{\geq 0}$ defined by $y_e := \frac{2}{5}$ for all $e \in M$ is feasible for (2ECM-LP) for G'. It then follows from the polyhedral analysis of Wolsey [16] and Shmoys and Williamson [15] of the (Subtour-LP) that we can find a 2-edge connected spanning multisubgraph of G' with edge set R satisfying $c(R) \leq \frac{3}{2}c^T y = \frac{3}{5}c(M)$. Then the graph H_1 induced by $F \cup R$ is a 2-edge connected spanning multisubgraph of G such that

$$c(H_1) \leq c(F) + \frac{3}{5}c(M) \leq c(F) + \frac{3}{5}c(E \setminus F) \ . \tag{9}$$

Now consider the vector $z \in \mathbb{R}^E_{\geq 0}$ where $z_e := 1/2$ for all $e \in F$, and $z_e := 1$ otherwise. Vector z is a feasible half-integer solution for (Subtour-LP), and thus by Theorem 2 and the ideas used in the proof of Theorem 1, in $O(n^2)$-time we can find a 2-edge connected spanning multisubgraph H_2 of G such that

$$c(H_2) \leq \frac{2}{3}c(F) + \frac{4}{3}c(E \setminus F). \tag{10}$$

We complete the proof by showing that either H_1 or H_2 has cost at most $\frac{7}{8}c(G)$. Using (9) and (10) we have:

$$\min(c(H_1), c(H_2)) \leq \frac{5}{8}c(H_1) + \frac{3}{8}c(H_2) \leq \frac{5}{8}\big(c(F) + \frac{3}{5}c(E \setminus F)\big) + \frac{3}{8}\big(\frac{2}{3}c(F) + \frac{4}{3}c(E \setminus F)\big) = \frac{7}{8}c(G).$$

◀

References

1. Anthony Alexander, Sylvia Boyd, and Paul Elliott-Magwood. On the integrality gap of the 2-edge connected subgraph problem. *Technical Report TR-2006-04, SITE, University of Ottawa, Ottawa, Canada*, 2006. URL: `http://www.site.uottawa.ca/~sylvia/publications/AlexanderBoydElliottmagwood2EC.pdf`.
2. Jørgen Bang-Jensen, Harold N. Gabow, Tibor Jordán, and Zoltán Szigeti. Edge-Connectivity Augmentation with Partition Constraints. *SIAM Journal on Discrete Mathematics*, 12(2):160–207, 1999. `doi:10.1137/S0895480197324700`.
3. Sylvia Boyd, Yao Fu, and Yu Sun. A 5/4-approximation for subcubic 2EC using circulations and obliged edges. *Discrete Applied Mathematics*, 209:48–58, 2016. `doi:10.1016/j.dam.2015.10.014`.
4. Sylvia Boyd, Satoru Iwata, and Kenjiro Takazawa. Finding 2-Factors Closer to TSP tours in Cubic Graphs. *SIAM Journal on Discrete Mathematics*, 27(2):918–939, 2013. `doi:10.1137/110843514`.
5. Sylvia Boyd and Philippe Legault. Toward a 6/5 Bound for the Minimum Cost 2-Edge Connected Spanning Subgraph. *SIAM Journal on Discrete Mathematics*, 31(1):632–644, 2017. `doi:10.1137/16M1057486`.
6. Robert Carr and R. Ravi. A new bound for the 2-edge connected subgraph problem. In Robert E. Bixby, E. Andrew Boyd, and Roger Z. Ríos-Mercado, editors, *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference*, volume 1412 of *Lecture Notes in Computer Science*, pages 112–125. Springer, Berlin, 1998. `doi:10.1007/3-540-69346-7_9`.
7. András Frank. *Connections in Combinatorial Optimization*. Oxford Lecture Series in Mathematics and its Applications, Volume 38. Oxford University Press, 2011.
8. Michel X Goemans and Dimitris J Bertsimas. Survivable networks, linear programming relaxations and the parsimonious property. *Mathematical Programming*, 60:145–166, 1993. `doi:10.1007/BF01580607`.
9. Arash Haddadan. *New Bounds on Integrality Gaps by Constructing Convex Combinations*. PhD Thesis. Carnegie Mellon University, 2020. URL: `https://www.cmu.edu/tepper/programs/phd/program/assets/dissertations/2020-joint-programs-algorithms-combinatorics-and-optimization-haddadan-dissertation.pdf`.
10. Arash Haddadan and Alantha Newman. Efficient constructions of convex combinations for 2-edge-connected subgraphs on fundamental classes. *CoRR*, abs/1811.09906, 2018. `arXiv:1811.09906`.
11. Arash Haddadan, Alantha Newman, and R. Ravi. Shorter tours and longer detours: uniform covers and a bit beyond. *Mathematical Programming*, 2019. `doi:10.1007/s10107-019-01426-8`.

12 Anna R. Karlin, Nathan Klein, and Shayan Oveis Gharan. An improved approximation algorithm for TSP in the half integral case. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proccedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 28–39. ACM, 2020. doi:10.1145/3357713.3384273.

13 Frans Schalekamp, David P. Williamson, and Anke van Zuylen. 2-Matchings, the Traveling Salesman Problem, and the Subtour LP: A Proof of the Boyd-Carr Conjecture. *Mathematics of Operations Research*, 39(2):403–417, 2014. doi:10.1287/moor.2013.0608.

14 András Sebő and Jens Vygen. Shorter tours by nicer ears: 7/5-Approximation for the graph-TSP, 3/2 for the path version, and 4/3 for two-edge-connected subgraphs. *Combinatorica*, 34(5):597–629, 2014. doi:10.1007/s00493-014-2960-3.

15 David B. Shmoys and David P. Williamson. Analyzing the Held-Karp TSP bound: a monotonicity property with application. *Information Processing Letters*, 35(6):281–285, 1990. doi:10.1016/0020-0190(90)90028-V.

16 Laurence A. Wolsey. Heuristic analysis, linear programming and branch and bound. *Mathematical Programming Study 13*, pages 121–134, 1980. doi:10.1007/BFb0120913.

Improved Multi-Pass Streaming Algorithms for Submodular Maximization with Matroid Constraints

Chien-Chung Huang
CNRS, DI ENS, Université PSL, Paris, France
chien-chung.huang@ens.fr

Theophile Thiery
School of Mathematical Sciences, Queen Mary University of London, UK
t.f.thiery@qmul.ac.uk

Justin Ward
School of Mathematical Sciences, Queen Mary University of London, UK
justin.ward@qmul.ac.uk

Abstract

We give improved multi-pass streaming algorithms for the problem of maximizing a monotone or arbitrary non-negative submodular function subject to a general p-matchoid constraint in the model in which elements of the ground set arrive one at a time in a stream. The family of constraints we consider generalizes both the intersection of p arbitrary matroid constraints and p-uniform hypergraph matching. For monotone submodular functions, our algorithm attains a guarantee of $p + 1 + \varepsilon$ using $O(p/\varepsilon)$-passes and requires storing only $O(k)$ elements, where k is the maximum size of feasible solution. This immediately gives an $O(1/\varepsilon)$-pass $(2 + \varepsilon)$-approximation for monotone submodular maximization in a matroid and $(3 + \varepsilon)$-approximation for monotone submodular matching. Our algorithm is oblivious to the choice ε and can be stopped after any number of passes, delivering the appropriate guarantee. We extend our techniques to obtain the first multi-pass streaming algorithms for general, non-negative submodular functions subject to a p-matchoid constraint. We show that a randomized $O(p/\varepsilon)$-pass algorithm storing $O(p^3 k \log(k)/\varepsilon^3)$ elements gives a $(p + 1 + \bar{\gamma}_{\text{off}} + O(\varepsilon))$-approximation, where $\bar{\gamma}_{\text{off}}$ is the guarantee of the best-known offline algorithm for the same problem.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Approximation algorithms analysis

Keywords and phrases submodular maximization, streaming algorithms, matroid, matchoid

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.62

Category APPROX

Funding *Chien-Chung Huang*: This work was funded by the grants ANR-19-CE48-0016 and ANR-18-CE40-0025-01 from the French National Research Agency (ANR).
Justin Ward: This work was supported by EPSRC New Investigator Award EP/T006781/1.

1 Introduction

Many discrete optimization problems in theoretical computer science, operations research, and machine learning can be cast as special cases of maximizing a *submodular* function f subject to some constraint. Formally, a function $f : 2^X \to \mathbb{R}_{\geq 0}$ is submodular if and only if $f(A) + f(B) \geq f(A \cup B) + f(A \cap B)$ for all $A, B \subseteq X$. One reason for the ubiquity of submodularity in optimization settings is that it also captures a natural "diminishing returns" property. Let $f(e \mid A) \triangleq f(A + e) - f(A)$ be the *marginal increase* obtained in f when adding an element e to a set A (where here and throughout we use the shorthands $A + e$


© Chien-Chung Huang, Theophile Thiery, and Justin Ward;
licensed under Creative Commons License CC-BY



and $A - e$ for $A \cup \{e\}$ and $A \backslash \{e\}$, respectively). Then it is well-known that f is submodular if and only if $f(e \mid B) \leq f(e \mid A)$ for any $A \subseteq B$ and any $e \notin B$. If additionally we have $f(e \mid A) \geq 0$ for all A and $e \notin A$ we say that f is *monotone*.

Here, we consider the problem of maximizing both monotone and arbitrary submodular functions subject to an arbitrary *p-matchoid constraint* on the set of elements that can be selected. Formally, a p-matchoid $\mathcal{M}^p = (\mathcal{I}^p, X)$ on X is given by a collection of matroids $\{\mathcal{M}_i = (X_i, \mathcal{I}_i)\}$ each defined on some subset of X, where each $e \in X$ is present in at most p of these subsets. A set $S \subseteq X$ is then independent if and only if $S \cap X_i \in \mathcal{I}_i$ for each matroid $\mathcal{M}_i$. One can intuitively think of a p-matchoid as a collection of matroids in which each element "participates" in at most p of the matroid constraints. The resulting family of constraints is quite general and captures both intersections of p matroid constraints (by letting $X_i = X$ for all $\mathcal{M}_i$) and matchings in p-uniform hypergraphs (by considering X as a collection of hyperedges and defining a uniform matroid constraint for each vertex, ensuring that at most one hyperedge containing this vertex is selected).

In many applications of submodular optimization, such as summarization [19, 1, 21, 23] we must process datasets so large that they cannot be stored in memory. Thus, there has been recent interest in *streaming* algorithms for submodular optimization problems. In this context, we suppose the ground set X is initially unknown and elements arrive one-by-one in a stream. We suppose that the algorithm has an efficient oracle for evaluating the submodular function f on any given subset of X, but has only enough memory to store a small number of elements from the stream. Variants of standard greedy and local search algorithms have been developed that obtain a constant-factor approximation in this setting, but their approximation guarantees are considerably worse than that of their simple, offline counterparts.

Here, we consider the *multi-pass* setting in which the algorithm is allowed to perform several passes over a stream – in each pass all of X arrives in some order, and the algorithm is still only allowed to store a small number of elements. In the *offline* setting, simple variants of greedy [15] or local search [18, 13] algorithms in fact give the best-known approximation guarantees for maximizing submodular functions subject to the p matroid constraints or a general p-matchoid constraint. However, these algorithms potentially require considering all elements in X *each time a choice is made*. It is natural to ask whether this is truly necessary, or whether we could instead recover an approximation ratio nearly equal to these offline algorithms by using only a constant number of passes through the data stream.

1.1 Our Results

Here we show that for monotone submodular functions, $O(1/\varepsilon)$-passes suffice to obtain guarantees only $(1 + \varepsilon)$ times worse than those guaranteed by the offline local search algorithm. We give an $O(p/\varepsilon)$-pass streaming algorithm that gives a $p + 1 + \varepsilon$ approximation for maximizing a monotone submodular function subject to an arbitrary p-matchoid constraint. This immediately gives us an $O(1/\varepsilon)$ pass streaming algorithm attaining a $2+\varepsilon$ approximation for matroid constraints and a $3 + \varepsilon$ approximation for matching constraints in graphs. Each pass of our algorithm is equivalent to a single pass of the streaming local search algorithm described by Chakrabarti and Kale [6] and Chekuri, Gupta, and Quanrud [7]. However, obtaining a rapid convergence to a $p + 1 + \varepsilon$ approximation requires some new insights. We show that if a pass makes either large or small progress in the value of f, then the guarantee obtained at the end of this pass can be improved. Balancing these two effects then leads a carefully chosen sequence of parameters for each pass. Our general approach is similar to that of Chakrabarti and Kale [6], but our algorithm is *oblivious* to the choice of ε. This

Table 1 Summary of known approximation ratios for maximizing monotone and non-monotone submodular functions. M means that f is monotone, and NN means that f is non-negative. In stating results parameterized by p, we let $o(1)$ denote a term that approaches zero as p increases.

Constraint	Offline	
	M	NN
matroid	$e/(e-1)$ [5]	2.598 [3]
rank p hypergraph b-matching	$p+\varepsilon$ [13]	$\frac{p^2+\varepsilon}{p-1}$ [13]
p matroid intersection	$p+\varepsilon$ [18]	$\frac{p^2+(p-1)\varepsilon}{p-1}$ [18]
p-matchoid	$p+1$ [15]	$\frac{ep}{(1-\varepsilon)(2-o(1))}$ [12, 8]

Constraint	Streaming		Multipass		
	M	NN	M	NN	#-passes
matroid	4 [6, 7, 11]	5.8284 [11]	$2+\varepsilon$ [6]		$O\left(\varepsilon^{-1}\right)$
rank p hypergraph b-matching	$4p$ [7, 11]	$4p+2-o(1)$ [11]	$p+1+\varepsilon$ [6]		$O\left(\frac{p^4\log(p)}{\varepsilon^3}\right)$
p matroid intersection	$4p$ [6, 7, 11]	$4p+2-o(1)$ [11]	$p+1+\varepsilon$ [6]		$O\left(\frac{p^4\log(p)}{\varepsilon^3}\right)$
p-matchoid	$4p$ [7, 11]	$4p+2-o(1)$ [11]			

allows us to give a uniform bound on the convergence of the approximation factor obtained after some number d of passes. This bound is actually available to the algorithm, and so we can certify the quality of the current solution after each pass. In practice, this allows for terminating the algorithm early if a sufficient guarantee has already been obtained. Even in the worst case, however, we improve on the number of passes required by similar previous results by a factor of $O(\varepsilon^{-2})$. Our algorithm only requires storing $O(k)$ elements, where k is the *rank* of the given p-matchoid, defined as the size of the largest independent set of elements.

Building on these ideas, we also give a randomized, multi-pass algorithm that uses $O(p/\varepsilon)$-passes and attains a $p+1+\bar{\gamma}_{\text{off}}+O(\varepsilon)$ approximation for maximizing an arbitrary submodular function subject to a p-matchoid constraint, where $\bar{\gamma}_{\text{off}}$ is the approximation ratio attained by best-known offline algorithm for the same problem. To the best of our knowledge, ours is the first multipass algorithm when the function is non-monotone. In this case, our algorithm requires storing $O(\frac{p^3 k\log k}{\varepsilon^3})$ elements. We remark that to facilitate comparison with existing work, we have stated all approximation guarantees as factors $\gamma\geq 1$. However, we note that if one states ratios of the form $1/\gamma$ less than 1, then our results lead to $1/\gamma-\varepsilon$ approximations in which all dependence on p can be eliminated (by setting simply selecting some $\varepsilon'=p\varepsilon$).

1.2 Related Work

There is a vast literature on submodular maximization with various constraints and different models of computation. In the offline model, the work on maximizing a monotone submodular function goes back to Nemhauser, Wolsey and Fischer [24]. Monotone submodular functions are well studied and many new and powerful results have been obtained since then. The best approximation algorithm under a matroid constraint is due to Calinescu et al. [5] which is the best that can be done using a polynomial number of queries [24] (if f is given as a value oracle) or assuming $P\neq NP$ [9] (if f is given explicitly). For more general constraints, Lee, Sviridenko and Vondrák obtained a $p+\varepsilon$ approximation algorithm under p matroid

Table 2 Summary of our results maximizing monotone and non-monotone submodular functions.

Constraint	Our results		
	monotone	non-negative	#-passes
matroid	$2+\varepsilon$	$4.598+\varepsilon$	$O\left(\frac{1}{\varepsilon}\right)$
rank p hypergraph b-matching	$p+1+\varepsilon$	$p+1+\frac{p^2}{p-1}+O(\varepsilon)$	$O\left(\frac{p}{\varepsilon}\right)$
p matroid intersection	$p+1+\varepsilon$	$p+1+\frac{p^2}{p-1}+O(\varepsilon)$	$O\left(\frac{p}{\varepsilon}\right)$
p-matchoid	$p+1+\varepsilon$	$p+1+\frac{ep}{(1-\varepsilon)(2-o(1))}+O(\varepsilon)$	$O\left(\frac{p}{\varepsilon}\right)$

constraints [18]. Feldman et al. [13] obtained the same approximation ratio for the general class of p-exchange systems. For general p-matchoid constraints, the best approximation ratio is $p+1$, which is attained by the standard greedy algorithm [15].

Non-monotone objectives are less understood even under the simplest assumptions. The current best-known result for maximizing a submodular function under a matroid constraint is 2.598 [3], which is far from the 2.093 hardness result [16]. Table 1 gives the best known bounds for the constraints that we consider in the paper.

Due to the large volume of data in modern applications, there has also been a line of research focused on developing *fast* algorithms for submodular maximization [2, 22]. However, all results we have discussed so far assume that the entire instance is available at any time, which may not be feasible for massive datasets. This has motivated the study of streaming submodular maximization algorithms with low memory requirements. Badaniyuru et al. [1] achieved a $2+\varepsilon$-approximation algorithm for maximizing a monotone submodular function under a cardinality constraint in the streaming setting. This was recently shown to be the best possible bound attainable in one pass with memory sublinear in the size of the instance [14]. Chakrabarti and Kale [6] gave a $4p$-approximation for the intersection of p matroid constraints or p-uniform hypergraph matching. Later, Chekuri et al. [7] generalized their argument to arbitrary p-matchoid constraints, and also gave a modified algorithm for handling non-monotone submodular objectives. A fast, randomized variant of the algorithm of [6] was studied by Feldman, Karbasi and Kazemi [11], who showed that it has the same approximation guarantee when f is monotone and achieves a $2p+2\sqrt{p(p+1)}+1=4p+2-o(1)$-approximation for general submodular function.

When multiple passes through the stream are allowed, less is known and the tradeoff between the approximation guarantee and the number of passes requires more attention. Assuming cardinality constraints, one can obtain a $\frac{e}{e-1}+\varepsilon$-multipass streaming algorithm in $O\left(\varepsilon^{-1}\right)$ passes (see [2, 17, 20, 21, 25]). Huang et al. [17] achieved a $2+\varepsilon$-approximation under a knapsack constraint in $O\left(\varepsilon^{-1}\right)$ passes. For the intersection of p partition matroids or rank p hypergraph matching, the number of passes becomes dependent on p. Chakrabarti and Kale [6][1] showed that if one allows $O\left(\frac{p^4\log(p)}{\varepsilon^3}\right)$-passes, a $p+1+\varepsilon$ approximation is possible. Here we show how to obtain the same guarantee for an arbitrary p-matchoid constraint, while reducing the number of passes to $O(p/\varepsilon)$.

[1] In [6] a bound of $O(\log p/\varepsilon^3)$ is stated. We note that there appears to be a small oversight in their analysis, arising from the fact that their convergence parameter κ in this case is $O(\varepsilon^3/p^4)$. In any case, it seems reasonable to assume that p is a small constant in most cases.

Algorithm 1 The multi-pass streaming local search algorithm.

procedure *MultipassLocalSearch*$(\alpha, \beta_1, \ldots, \beta_d)$
  $S_0 \leftarrow \emptyset$;
  for $i = 1$ **to** d **do**
    Let $\tilde{S}$ be the output of StreamingLocalSearch$(\alpha, \beta_i, S_{i-1})$;
    $S_i \leftarrow \tilde{S}$;
  return S_d;

procedure *StreamingLocalSearch*$(\alpha, \beta, S_{\text{init}})$
  $S \leftarrow S_{\text{init}}$;
  foreach x in the stream **do**
    if $x \in S_{\text{init}}$ **then** discard x;
    $C_x \leftarrow$ Exchange(x, S);
    if $f(x|S) \geq \alpha + (1+\beta)\sum_{c \in C_x} \nu(c, S)$ **then**
      $S \leftarrow S \backslash C_x + x$;
  return S;

Algorithm 2 The procedure Exchange(x, S).

procedure *Exchange*(x, S)
  $C_x \leftarrow \emptyset$;
  foreach $\mathcal{M}_\ell = (X_\ell, \mathcal{I}_\ell)$ with $x \in X_\ell$ **do**
    $S_\ell \leftarrow S \cap X_\ell$;
    if $S_\ell + x \notin \mathcal{I}$ **then**
      $T_\ell \leftarrow \{y \in S_\ell : S_\ell - y + x \in \mathcal{I}_\ell\}$;
      $C_x \leftarrow C_x + \arg\min_{t \in T_\ell} \nu(t, S)$;
  return C_x;

2 The main multi-pass streaming algorithm

Our main multi-pass algorithm is given by the procedure MultipassLocalSearch in Algorithm 1. We suppose that we are given a submodular function $f : 2^X \to \mathbb{R}_{\geq 0}$ and a p-matchoid constraint $\mathcal{M}^p = (\mathcal{I}^p, X)$ on X given as a collection of matroids $\{\mathcal{M}_i = (X_i, \mathcal{I}_i)\}$. Our procedure runs for d passes, each of which uses a modification of the algorithm of Chekuri, Gupta, and Quanrud [7], given as the procedure StreamingLocalSearch. In each pass, procedure StreamingLocalSearch maintains a current solution S, which is initially set to some S_{init}. Whenever an element $x \in S_{\text{init}}$ arrives again in the subsequent stream, the procedure simply discards x. For all other elements x, the procedure invokes a helper procedure Exchange, given formally in Algorithm 2, to find an appropriate set $C_x \subseteq S$ of up to p elements so that $S \backslash C_x + x \in \mathcal{I}$. It then exchanges x with C_x if this gives a significantly improved solution. This improvement is measured with respect to a set of auxiliary weights $\nu(x, S)$ maintained by the algorithm. For $u, v \in X$, let $u \prec v$ denote that "element u arrives before v" in the stream. Then, we define the *incremental value* of an element e with respect to a set T as

$$\nu(e, T) = f(e \mid \{t' \in T : t' \prec e\}) .$$

There is a slight difficulty here in that we must also define incremental values for the elements of S_{init}. To handle this difficulty, we in fact define $\prec$ with respect to a *pretend* stream ordering. Note that in all invocations of the procedure StreamingLocalSearch made by MultipassLocalSearch, the set S_{init} is either $\emptyset$ or the result of a previous application of StreamingLocalSearch. In our pretend ordering ($\prec$) all of S_{init} first arrives in the same relative pretend ordering as the previous pass, followed by all of $X \backslash S_{\text{init}}$ in the same order given by the stream X. We then define our incremental values with respect to this pretend stream ordering.

Using these incremental values, StreamingLocalSearch proceeds as follows. When an element $x \notin S_{\text{init}}$ arrives, StreamingLocalSearch computes a set of elements $C_x \subseteq S$ that can be exchanged for x. StreamingLocalSearch replaces C_x with x if and only if the marginal value $f(x \mid S)$ with respect to S is at least $(1+\beta)$ times larger than the sum of the current incremental values $\nu(c, S)$ of all elements $c \in C_x$ plus some threshold α, where $\alpha, \beta > 0$ are given as parameters. In this case, we say that the element x is *accepted*. Otherwise, we say that x is *rejected*. An element $x \in S$ that has been accepted may later be removed from S if $x \in C_y$ for some later element y that arrives in the stream. In this case we say that x is *evicted*.

The approximation ratio obtained by one pass of StreamingLocalSearch depends on the parameter β in two ways, which can be intuitively understood in terms of the standard analysis of the offline local search algorithm for the problem. Intuitively, if β is chosen to be too large, more valuable elements will be rejected upon arrival and so, in the offline setting, our solution would be only approximately locally optimal, leading to a deterioration of the guarantee by a factor of $(1+\beta)$. However, in the streaming setting, the algorithm only attempts to exchange an element upon its arrival, and so the final solution will not necessarily be even $(1+\beta)$-approximately locally optimal – an element x may be rejected because $f(x \mid S)$ is small when it arrives, but the processing of later elements in the stream can evict some elements of S. After these evictions, we could have $f(x \mid S)$ larger. The key observation in the analyses of [6, 7] is that the total value of these evicted elements – and so also the total increase in the marginal value of all rejected elements – can be bounded by $O(\frac{1}{\beta})$ times the final value of $f(S)$ at the end of the algorithm. Intuitively, if β is chosen to be too small, the algorithm will make more exchanges, evicting more elements, which may result in rejected elements being much more valuable with respect to the final solution. Selecting the optimal value of β thus requires balancing these two effects.

Here, we observe that this second effect depends only on the total value of those elements that were accepted *after* an element arrives. To use this observation, we measure the ratio $\delta = f(S_{\text{init}})/f(\tilde{S})$ between the value of the initial solution S_{init} of some pass of StreamingLocalSearch and the final solution $\tilde{S}$ produced by this pass. If δ is relatively small – and so one pass makes a lot of progress – then this pass gives us an improvement of δ^{-1} over the ratio already guaranteed by the previous pass since $f(\tilde{S}) = \delta^{-1} f(S_{\text{init}})$. On the other hand, if δ is relatively large – and so one pass does not make much progress – then the total increase in the value of our rejected elements can be bounded by $\frac{1-\delta}{\beta} f(\tilde{S})$, and so the potential loss due to only testing these elements at arrival is relatively small. Balancing these two effects allows us to set β smaller in each subsequent passes and obtain an improved guarantee.

We now turn to the analysis of our algorithm. Here we focus on a single pass of StreamingLocalSearch. For $T, U \subseteq X$ we let $f(T \mid U) \triangleq f(T \cup U) - f(U)$. Throughout, we use S to denote the current solution maintained by this pass (initially, $S = S_{\text{init}}$). The following key properties of incremental values will be useful in our analysis. We defer the proof to the Appendix.

▶ **Lemma 1.** *For any $T \subseteq U \subseteq X$,*
1. $\sum_{e \in T} \nu(e, T) = f(T) - f(\emptyset)$
2. $\nu(e, U) \leq \nu(e, T)$ *for all* $e \in T$.
3. $f(T \mid U \backslash T) \leq \sum_{t \in T} \nu(t, U)$
4. *At all times during the execution of* StreamingLocalSearch, $\nu(e, S) \geq \alpha$ *for all* $e \in S$.

Let A denote the set of elements *accepted* during the present pass. These are the elements which were present in the solution S at some previous time during the execution of this pass. Initially we have $A = S = S_{\text{init}}$ and whenever an element is added to S, during this pass we also add this element to A. Let $\tilde{A}$ and $\tilde{S}$ denote the sets of elements A and S at the end of this pass. Note that we regard all elements of S_{init} as having been accepted at the start of the pass. The following lemma follows from the analysis of Chekuri, Gupta, and Quanrud [7] in the single-pass setting. We give a complete, self-contained proof in Appendix A. Each element $e \in \tilde{A} \backslash \tilde{S}$ was accepted but later *evicted* by the algorithm. For any such evicted element, we let $\chi(e)$ denote the value of $\nu(e, S)$ at the moment that e was removed from S.

▶ **Lemma 2.** *Let $f : 2^X \to \mathbb{R}_{\geq 0}$ be a submodular function. Suppose $\tilde{S}$ is the solution produced at the end of one pass of* StreamingLocalSearch *and $\tilde{A}$ be the set of all elements accepted during this pass. Then,*

$$f(OPT \cup \tilde{A}) \leq (p + \beta p - \beta) \sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e) + (p + \beta p + 1) f(\tilde{S}) + k\alpha \,.$$

We now derive a bound for the summation $\sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e)$ (representing the value of *evicted* elements) in terms of the total gain $f(\tilde{S}) - f(S_{\text{init}})$ made by the pass, and also bound the total number of accepted elements in terms of $f(OPT)$.

▶ **Lemma 3.** *Let $f : 2^X \to \mathbb{R}_{\geq 0}$ be a submodular function. Suppose that $\tilde{S}$ is the solution produced at the end of one pass of* StreamingLocalSearch *and $\tilde{A}$ is the set of all elements accepted during this pass. Then, $|\tilde{A}| \leq f(OPT)/\alpha$ and*

$$\sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e) \leq \frac{1}{\beta} \left(f(\tilde{S}) - f(S_{\text{init}}) \right) .$$

Proof. We consider the quantity $\Phi(A) \triangleq \sum_{e \in A \backslash S} \chi(e)$. Suppose some element a with $C_a \neq \emptyset$ is added to S by the algorithm, evicting the elements of C_a. Then (as each element can be evicted only once) $\Phi(A)$ increases by precisely $\Delta \triangleq \sum_{e \in C_a} \chi(e)$. Let S_a^-, S_a^+ and A_a^-, A_a^+ be the sets S and A, respectively, immediately before and after a is accepted. Since a is accepted, we must have $f(a \mid S_a^-) \geq \alpha + (1 + \beta) \sum_{e \in C_a} \nu(e, S_a^-)$. Then,

$$\begin{aligned}
f(S_a^+) - f(S_a^-) &= f(S_a^- \backslash C_a + a) - f(S_a^-) \\
&= f(a \mid S_a^- \backslash C_a) - f(C_a \mid S_a^- \backslash C_a) \\
&\geq f(a \mid S_a^-) - f(C_a \mid S_a^- \backslash C_a) && \text{(by submodularity)} \\
&\geq f(a \mid S_a^-) - \sum_{e \in C_a} \nu(e, S_a^-) && \text{(by Lemma 1 (3))} \\
&\geq \alpha + (1 + \beta) \sum_{e \in C_a} \nu(e, S_a^-) - \sum_{e \in C_a} \nu(e, S_a^-) && \text{(since } a \text{ is accepted)} \\
&= \alpha + \beta \sum_{e \in C_a} \chi(e) \,. && \text{(by definition of } \chi(e)) \\
&= \alpha + \beta \Delta
\end{aligned}$$

It follows that whenever $\Phi(A)$ increases by Δ, $f(S)$ must increase by at least $\beta\Delta$. Initially, $\Phi(A) = 0$ and $f(S) = f(S_{\text{init}})$ and at the end of the algorithm, $\Phi(A) = \sum_{e\in\tilde{A}\setminus\tilde{S}} \chi(e)$ and $f(S) = f(\tilde{S})$. Thus, $\beta \sum_{e\in\tilde{A}\setminus\tilde{S}} \chi(e) \leq [f(\tilde{S}) - f(S_{\text{init}})]$.

It remains to show that $|\tilde{A}| \leq f(OPT)/\alpha$. For this, we note that the above chain of inequalities also implies that every time an element is accepted (and so $|A|$ increases by one), $f(S)$ also increases by at least α. Thus, we have $f(OPT) \geq f(\tilde{S}) \geq \alpha|\tilde{A}|$. ◀

Using Lemma 3 to bound the sum of exit values in Lemma 2 then immediately gives us the following guarantee for each pass performed in MultipassLocalSearch. In the i^{th} such pass, we will have $S_{\text{init}} = S_{i-1}$, $\tilde{S} = S_i$, and $\beta = \beta_i$. We let A_i denote the set of $\tilde{A}$ of all elements accepted during this particular pass.

▶ **Lemma 4.** *Let $f : 2^X \to \mathbb{R}_{\geq 0}$ be a submodular function. Consider the i^{th} pass of StreamingLocalSearch performed by MultipassLocalSearch, and let A_i be the set of all elements accepted during this pass. Then, $|A_i| \leq f(OPT)/\alpha$ and*

$$f(OPT \cup A_i) \leq (p/\beta_i + p - 1)\,[f(S_i) - f(S_{i-1})] + (p + p\beta_i + 1)f(S_i) + k\alpha\,.$$

3 Analysis of the multipass algorithm for monotone functions

We now show how to use Lemma 4 together with a careful selection of parameters α and $\beta_1, \ldots, \beta_d$ to derive guarantees for the solution $f(S_i)$ produced after the i^{th} pass made in MultipassLocalSearch. Here, we consider the case that f is a *monotone* function. In this case, we have $f(OPT) \geq f(OPT \cup A_i)$ for all i. We set $\alpha = 0$ in each pass. In the first pass, we will set $\beta_1 = 1$. Then, since $S_0 = \emptyset$ Lemma 4 immediately gives:

$$f(OPT) \leq f(OPT \cup A_1) \leq (2p - 1)\,[f(S_1) - f(\emptyset)] + (2p + 1)f(S_1) = 4pf(S_1)\,. \tag{1}$$

For passes $i > 1$, we use the following, which relates the approximation guarantee obtained in this pass to that from the previous pass.

▶ **Theorem 5.** *For $i > 1$, suppose that $f(OPT) \leq \gamma_{i-1} \cdot f(S_{i-1})$ and define $\delta_i = \frac{f(S_{i-1})}{f(S_i)}$. Then,*

$$f(OPT) \leq \min\left\{\gamma_{i-1}\delta_i, (\tfrac{p}{\beta_i} + p - 1)(1 - \delta_i) + p + \beta_i p + 1\right\} \cdot f(S_i) + k\alpha\,.$$

Proof. From the definition of γ_{i-1} and δ_i, we have:

$$f(OPT) \leq \gamma_{i-1} f(S_{i-1}) = \gamma_{i-1}\delta_i f(S_i)\,.$$

On the other hand, $f(S_i) - f(S_{i-1}) = (1 - \delta_i)f(S_i)$. Thus, Lemma 4 gives:

$$f(OPT) \leq [(p/\beta_i + p - 1)\,(1 - \delta_i) + p + \beta_i p + 1]\,f(S_i) + k\alpha\,.$$ ◀

Now, we observe that for any fixed guarantee γ_{i-1} from the previous pass, $\gamma_{i-1}\delta_i$ is an increasing function of δ_i and $((p/\beta_i + p - 1)(1 - \delta_i) + p + \beta_i p + 1)$ is an decreasing function of δ_i. Thus, the guarantee we obtain in Theorem 5 is always at least as good as that obtained when these two values are equal. Setting:

$$\gamma_{i-1}\delta_i = (\tfrac{p}{\beta_i} + p - 1)(1 - \delta_i) + p + \beta_i p + 1,$$

and solving for δ_i gives us:

$$\delta_i = \frac{p(1 + \beta_i)^2}{p + \beta_i(\gamma_{i-1} - 1 + p)}\,. \tag{2}$$

In the following analysis, we consider this value of δ_i since the guarantee given by Theorem 5 will always be no worse than that given by this value. The analysis for a single matroid constraint follows from our results for p-matchoids, but the analysis and parameter values obtained are much simpler, so we present it separately, first.

▶ **Theorem 6.** *Suppose we run Algorithm 1 for an arbitrary matroid constraint and monotone submodular function f, with $\beta_i = \frac{1}{i}$. Then $2(1+\frac{1}{i})f(S_i) \geq f(OPT)$ for all $i > 0$. In particular, after $i = \frac{2}{\varepsilon}$ passes, $(2+\varepsilon)f(S_i) \geq f(OPT)$.*

Proof. Let γ_i be the guarantee for our algorithm after i passes. We show, by induction on i, that $\gamma_i \leq \frac{2(i+1)}{i}$. For $i = 1$, we have $\beta_1 = 1$ and so from (1) we have $\gamma_1 = 4$, as required. For $i > 1$, suppose that $\gamma_{i-1} \leq \frac{2i}{i-1}$. Since $p = 1$ and $\beta_i = 1/i$ (2) gives:

$$\delta_i \leq \frac{(1+\frac{1}{i})^2}{1+\frac{1}{i}(\frac{2i}{i-1})} = \frac{\frac{(i+1)^2}{i^2}}{\frac{(i-1)+2}{i-1}} = \frac{(i-1)(i+1)}{i^2}\,.$$

Thus, by Theorem 5, the i^{th} pass of our algorithm has guarantee γ_i satisfying:

$$\gamma_i \leq \gamma_{i-1}\delta_i \leq \frac{2i}{i-1}\frac{(i-1)(i+1)}{i^2} = \frac{2(i+1)}{i}\,,$$

as required. ◀

▶ **Theorem 7.** *Suppose we run Algorithm 1 for an arbitrary p-matchoid constraint and monotone submodular function f, β_1 and*

$$\beta_i = \frac{\gamma_{i-1}-1-p}{\gamma_{i-1}-1+p}$$

for $i > 1$, where γ_i is given by the recurrence $\gamma_1 = 4p$ and

$$\gamma_i = 4p\frac{\gamma_{i-1}(\gamma_{i-1}-1)}{(\gamma_{i-1}-1+p)^2}$$

for $i > 1$. Then $\left(p+1+\frac{4p}{i}\right)f(S_i) \geq f(OPT)$ for all $i > 0$. In particular, after $i = \frac{4p}{\varepsilon}$ passes, $(p+1+\varepsilon)f(S_i) \geq f(OPT)$.

Proof. We first show that approximation guarantee of our algorithm after i passes is given by γ_i. Setting $\beta_1 = 1$, we obtain $\gamma_1 = 4p$ from (1), agreeing with our definition. For passes $i > 1$, let $\beta_i = \frac{\gamma_{i-1}-1-p}{\gamma_{i-1}-1+p}$. As in the case of matroid constraint, Theorem 5 implies that the guarantee for pass i will be at most $\delta_i\gamma_{i-1}$, where δ_i is chosen to satisfy (2). Specifically, if we set

$$\delta_i = \frac{p\left(1+\frac{\gamma_{i-1}-1-p}{\gamma_{i-1}-1+p}\right)^2}{p+\frac{\gamma_{i-1}-1-p}{\gamma_{i-1}-1+p}(\gamma_{i-1}-1+p)} = \frac{p\left(\frac{2(\gamma_{i-1}-1)}{\gamma_{i-1}-1+p}\right)^2}{\gamma_{i-1}-1} = \frac{4p(\gamma_{i-1}-1)}{(\gamma_{i-1}-1+p)^2}\,,$$

then we have $\delta_i\gamma_{i-1} = \gamma_i$.

We now show by induction on i that $\gamma_i \leq p+1+\frac{4p}{i}$. In the case $i = 1$, we have $\gamma_1 = 4p$ and the claim follows immediately from $p \geq 1$. In the general case $i > 0$, and we may assume without loss of generality that $\gamma_{i-1} \geq 1$. Otherwise the theorem holds immediately, as each subsequent pass can only increase the value of the solution. Then, we note (as shown in Appendix B) that for $p \geq 1$ and $\gamma_{i-1} \geq 1$, γ_i is an increasing function of γ_{i-1}. By the induction hypothesis, $\gamma_{i-1} \leq p+1+\frac{4p}{i-1}$. Therefore:

$$\gamma_i \leq \frac{4p\left(p+1+\frac{4p}{i-1}\right)\left(p+\frac{4p}{i-1}\right)}{\left(2p+\frac{4p}{i-1}\right)^2} \leq p+1+\frac{4p}{i}\,,$$

as required. The last inequality above follows from straightforward but tedious algebraic manipulations, which can be found in Appendix B. ◀

Algorithm 3 The randomized multi-pass streaming algorithm.

procedure MultipassRandomizedLocalSearch($\alpha, \beta_1, \ldots, \beta_d, m$)
  $S_0 \leftarrow \emptyset, S'_0 \leftarrow \emptyset$;
  for $i = 1$ **to** d **do**
    Let $(\tilde{S}, S')$ be the output of RandomizedLocalSearch($S_{i-1}, \alpha, \beta_i, m$);
    $S_i \leftarrow \tilde{S}$, $S'_i \leftarrow \arg\max\{f(S'_{i-1}), f(S')\}$;
  return $\bar{S} = \arg\max\{f(S_d), f(S'_d)\}$;

procedure RandomizedLocalSearch($S_{\text{init}}, \alpha, \beta, m$)
  $S \leftarrow S_{\text{init}}$; $B \leftarrow \emptyset$;
  foreach x in the stream **do**
    if $f(x \mid S) \geq \alpha + (1+\beta)\sum_{e \in C_x} \nu(e, S)$ **then**
      $B \leftarrow B + x$;
    if $|B| = m$ **then**
      $x \leftarrow$ uniformly random element from B;
      $C_x \leftarrow$ Exchange(x, S);
      $B \leftarrow B - x$; $S \leftarrow S + x - C_x$;
      foreach x' in B **do**
        $C_{x'} \leftarrow$ Exchange(x', S);
        if $f(x' \mid S) < \alpha + (1+\beta)\sum_{e \in C_{x'}} \nu(e, S)$ **then**
          $B \leftarrow B - x'$;
  $S' \leftarrow$ Offline(B);
  return (S, S');

4 A multi-pass algorithm for general submodular functions

In this section, we show that the guarantees for monotone submdodular maximization can be extended to non-monotone submodular maximization even when dealing with multiple passes. Our main algorithm is given by procedure MultipassRandomizedLocalSearch in Algorithm 3. In each pass, it calls a procedure RandomizedLocalSearch, which is an adaptation of StreamingLocalSearch, to process the stream. Note that each such pass produces a pair of feasible solutions S and S', which we now maintain throughout MultipassRandomizedLocalSearch. The set S is maintained similarly as before and gradually improves by exchanging "good" elements into a solution throughout the pass. The set S' will be maintained by considering the best output of an offline algorithm that we run after each pass as described in more detail below.

To deal with non-monotone submodular functions, we will limit the probability of elements being added to S. Instead of exchanging good elements on arrival, we store them in a buffer B of size m. When the buffer becomes full, an element is chosen uniformly at random and added to S. Adding a new element to the current solution may affect the quality of the remaining elements in the buffer and thus we need to re-evaluate them and remove the elements that are no longer good. As before, we let A denote the set of elements that were previously added to S during the current pass of the algorithm. Note that we do not consider an element to be accepted until it has actually been added to S from the buffer. For any fixed set of random choices, the execution of RandomizedLocalSearch can be considered

as the execution of StreamingLocalSearch on the following stream: we suppose that an element x arrives whenever it is selected from the buffer and accepted into S. All elements that are discarded from the buffer after accepting x then arrive, and will also be rejected by StreamingLocalSearch. Any elements remaining in the buffer after the execution of the algorithm do not arrive in the stream. Applying Lemma 4 with respect to this pretend stream ordering allows us to bound $f(\tilde{S})$ with respect to $f(OPT \setminus B)$ (that is, the value of the part of OPT that does not remain in the buffer B) after a single pass of RandomizedLocalSearch. Formally, let $\tilde{B}_i$ be the value of the buffer after the i^{th} pass of our algorithm. Then, applying Lemma 4 to the set $OPT \setminus \tilde{B}_i$, and taking expectation, gives:

$$\mathbb{E}[f(A_i \cup (OPT \setminus \tilde{B}_i))] \leq (p/\beta + p - 1)\,(\mathbb{E}[f(S_i)] - \mathbb{E}[f(S_{i-1})]) + (p + \beta p + 1)\,\mathbb{E}[f(S_i)] + \alpha k\,. \quad (3)$$

In order to bound the value of the elements in $\tilde{B}_i$, we apply any offline $\bar{\gamma}_{\text{off}}$-approximation algorithm Offline to the buffer at the end of the pass to obtain a solution S'. In MultipassRandomizedLocalSearch, we then remember the best such offline solution S'_i computed across the first i passes. Then, in the i^{th} pass, we have

$$\mathbb{E}[f(OPT \cap \tilde{B}_i)] \leq \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S')] \leq \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)]\,. \quad (4)$$

From submodularity of f and $A_i \cap \tilde{B}_i = \emptyset$ we have $f(A_i \cup OPT) \leq f(A_i \cup (OPT \setminus \tilde{B}_i)) + f(OPT \cap \tilde{B}_i)$. Thus, combining (3) and (4) we have:

$$\mathbb{E}[f(A_i \cup OPT)] \leq (p/\beta + p - 1)\,(\mathbb{E}[f(S_i)] - \mathbb{E}[f(S_{i-1})]) + (p + \beta p + 1)\,\mathbb{E}[f(S_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)] + \alpha k\,. \quad (5)$$

To relate the right-hand side to $f(OPT)$ we use the following result from Buchbinder et al. [4]:

▶ **Lemma 8** (Lemma 2.2 in [4]). *Let $f\colon 2^X \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Suppose that A is a random set where no element $e \in X$ appears in A with probability more than p. Then, $\mathbb{E}[f(A)] \geq (1-p)\,f(\emptyset)$. Moreover, for any set $Y \subseteq X$, it follows that $\mathbb{E}[f(Y \cup A)\,] \geq (1-p) f(Y)$.*

We remark that a similar theorem also appeared earlier in Feige, Mirrokni, and Vondrák [10] for a random set that contains each element *independently* with probability *exactly* p. Here, the probability that an element occurs in A_i is delicate to handle because such an element may either originate from the starting solution S_{i-1} or be added during the pass. Thus, we use a rougher estimate. By definition $A_i \subseteq A_i \cup A_{i-1} \cup \ldots \cup A_1$. Thus, $\Pr[e \in A_i] \leq \Pr[e \in A_i \cup \ldots \cup A_1]$. The number of selections during the j^{th} pass is at most $|A_j|$ and by Lemma 4 (applied to the set $OPT \setminus \tilde{B}_j$ due to our pretend stream ordering in each pass j), $|A_j| \leq f(OPT \setminus \tilde{B}_j)/\alpha \leq f(OPT)/\alpha$ in any pass. Here, the second inequality follows from the optimality of OPT, and the fact that any subset of the feasible solution OPT is also feasible for our p-matchoid constraint. Thus, the total number of selections in the first i passes at most $\sum_{j=1}^{i} |A_j| \leq i \cdot f(OPT)/\alpha$. We select an element only when the buffer is full, and each selection is made independently and uniformly at random from the buffer. Thus, the probability that any given element is selected when the algorithm makes a selection is at most $1/m$ and by a union bound, $\Pr[e \in A_i \cup \ldots \cup A_1] \leq i \cdot f(OPT)/(m\alpha)$. Let d be the number of passes that the algorithm makes and suppose we set $\alpha = \varepsilon f(OPT)/2k$ (in Appendix C we show that this can be accomplished approximately by guessing $f(OPT)$, which can be done at the expense of an extra factor $O(\log k)$ space). Finally, let $m = 4dk/\varepsilon^2$. Then, applying Lemma 8, after $i \leq d$ passes we have:

$$\mathbb{E}[f(A_i \cup OPT)] \geq (1 - d \cdot f(OPT)/(m\alpha))\,f(OPT) \geq (1 - \varepsilon/2)\,f(OPT)\,. \quad (6)$$

Our definition of α also implies that $\alpha k \leq (1-\varepsilon/2)f(OPT)$. Using this and (6) in (5), we obtain:

$$(1-\varepsilon)f(OPT) \leq (p/\beta + p - 1)(\mathbb{E}[f(S_i)] - \mathbb{E}[f(S_{i-1})]) + (p + \beta p + 1)\,\mathbb{E}[f(S_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)]\,. \quad (7)$$

As we show in Appendix C, the rest of the analysis then follows similarly to that in Section 3, using the fact that $f(\bar{S}) = \max\{f(S_d), f(S'_d)\}$.

▶ **Theorem 9.** *Let $\mathcal{M}^p = (X, \mathcal{I})$ be a p-matchoid of rank k and let $f\colon 2^X \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Suppose there exists an algorithm for the offline instance of the problem with approximation factor $\bar{\gamma}_{\text{off}}$. For any $\varepsilon > 0$, the randomized streaming local-search algorithm returns a solution $\bar{S} \in \mathcal{I}$ such that*

$$f(OPT) \leq (p + 1 + \bar{\gamma}_{\text{off}} + O(\varepsilon))\,\mathbb{E}[f(\bar{S})]$$

using a total space of $O\Big(\frac{p^3 k \log_2 k}{\varepsilon^3}\Big)$ and $O\big(\frac{p}{\varepsilon}\big)$-passes.

References

1 Ashwinkumar Badanidiyuru, Baharan Mirzasoleiman, Amin Karbasi, and Andreas Krause. Streaming submodular maximization: massive data summarization on the fly. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 671–680, 2014.

2 Ashwinkumar Badanidiyuru and Jan Vondrák. Fast algorithms for maximizing submodular functions. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1497–1514, 2013.

3 Niv Buchbinder and Moran Feldman. Constrained submodular maximization via a nonsymmetric technique. *Mathematics of Operations Research*, 44(3):988–1005, 2019.

4 Niv Buchbinder, Moran Feldman, Joseph Naor, and Roy Schwartz. Submodular maximization with cardinality constraints. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1433–1452, 2014.

5 Gruia Calinescu, Chandra Chekuri, Martin Pál, and Jan Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

6 Amit Chakrabarti and Sagar Kale. Submodular maximization meets streaming: matchings, matroids, and more. *Mathematical Programming*, 154(1-2):225–247, 2015.

7 Chandra Chekuri, Shalmoli Gupta, and Kent Quanrud. Streaming algorithms for submodular function maximization. In *Proc. International Colloquium on Automata, Languages, and Programming (ICALP)*, pages 318–330, 2015.

8 Chandra Chekuri, Jan Vondrák, and Rico Zenklusen. Submodular function maximization via the multilinear relaxation and contention resolution schemes. *SIAM Journal on Computing*, 43(6):1831–1879, 2014.

9 Uriel Feige. A threshold of $\ln n$ for approximating set cover. *Journal of the ACM*, 45(4):634–652, 1998.

10 Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

11 Moran Feldman, Amin Karbasi, and Ehsan Kazemi. Do less, get more: streaming submodular maximization with subsampling. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 732–742, 2018.

12 Moran Feldman, Joseph Naor, and Roy Schwartz. A unified continuous greedy algorithm for submodular maximization. In *Proc. IEEE Symposium on Foundations of Computer Science, (FOCS)*, pages 570–579, 2011.

13 Moran Feldman, Joseph (Seffi) Naor, Roy Schwartz, and Justin Ward. Improved approximations for k-exchange systems. In *Proc. European Symposium on Algorithms (ESA)*, pages 784–798, 2011.

14 Moran Feldman, Ashkan Norouzi-Fard, Ola Svensson, and Rico Zenklusen. The one-way communication complexity of submodular maximization with applications to streaming and robustness. In *Proc. ACM Symposium on Theory of Computing (STOC)*, pages 1363–1374, 2020.

15 M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions II. *Mathematical Programming Study*, 8:73–87, 1978.

16 Shayan Oveis Gharan and Jan Vondrák. Submodular maximization by simulated annealing. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1098–1116, 2011.

17 Chien-Chung Huang and Naonori Kakimura. Multi-pass streaming algorithms for monotone submodular function maximization. *CoRR*, abs/1802.06212, 2018. `arXiv:1802.06212`.

18 Jon Lee, Maxim Sviridenko, and Jan Vondrák. Submodular maximization over multiple matroids via generalized exchange properties. *Mathematics of Operations Research*, 35(4):795–806, 2010.

19 Hui Lin and Jeff Bilmes. Multi-document summarization via budgeted maximization of submodular functions. In *Proc. Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 912–920, 2010.

20 Andrew McGregor and Hoa T. Vu. Better streaming algorithms for the maximum coverage problem. *Theory of Computing Systems*, 63(7):1595–1619, 2019.

21 Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, and Aamin Karbasi. Fast constrained submodular maximization: Personalized data summarization. In *Proc. International Conference on Machine Learning (ICML)*, pages 1358–1367, 2016.

22 Baharan Mirzasoleiman, Ashwinkumar Badanidiyuru, Amin Karbasi, Jan Vondrák, and Andreas Krause. Lazier than lazy greedy. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 1812–1818, 2015.

23 Baharan Mirzasoleiman, Stefanie Jegelka, and Andreas Krause. Streaming non-monotone submodular maximization: Personalized video summarization on the fly. In *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, pages 1379–1386, 2018.

24 G L Nemhauser and L A Wolsey. Best algorithms for approximating the maximum of a submodular set function. *Mathematics of Operations Research*, 3(3):177–188, 1978.

25 Ashkan Norouzi-Fard, Jakub Tarnawski, Slobodan Mitrović, Amir Zandieh, Aida Mousavifar, and Ola Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proc. International Conference on Machine Learning (ICML)*, pages 3826–3835, 2018.

A Proof of Lemma 2

Here, we give a self-contained analysis of the single-pass algorithm of Chekuri, Gupta, and Quanrud [7], corresponding to Algorithm 1 initialized with $S_{\text{init}} = \emptyset$. First, we prove Lemma 1, which concerns properties of the incremental values maintained by Algorithm 1.

▶ **Lemma 1.** *For any $T \subseteq U \subseteq X$,*

1. $\sum_{e \in T} \nu(e, T) = f(T) - f(\emptyset)$
2. $\nu(e, U) \leq \nu(e, T)$ *for all* $e \in T$.
3. $f(T \mid U \backslash T) \leq \sum_{t \in T} \nu(t, U)$
4. *At all times during the execution of* StreamingLocalSearch, $\nu(e, S) \geq \alpha$ *for all* $e \in S$.

Proof. Property (1) follows directly from the telescoping summation

$$\sum_{e\in T}\nu(e,T)=\sum_{e\in T}[f(e\cup\{t'\in T:t'\prec e\})-f(\{t'\in T:t'\prec e\}]=f(T)-f(\emptyset).$$

Property (2) follows from submodularity since $T\subseteq U$ implies that $\{t'\in T:t'\prec e\}\subseteq\{t'\in U:t'\prec e\}$.

For property (3), we note that:

$$f(T\mid U\backslash T)=\sum_{t\in T}f(t\mid U\backslash T\cup\{t'\in T:t'\prec t\})\leq\sum_{t\in T}f(t\mid\{u'\in U:u'\prec t\})=\sum_{t\in T}\nu(t,U)\,,$$

where the first equation follows from a telescoping summation, and the inequality follows from submodularity, since $\{u'\in U:u'\prec t\}\subseteq U\setminus T\cup\{t'\in T:t'\prec t\}$.

We prove property (4) by induction on the stream of elements arriving. Initially $S=\emptyset$. Thus, the first time that any element x is accepted, we must have $C_x=\emptyset$ and so $f(x\mid S)\geq\alpha$. After this element is accepted, we have $\nu(x,S)=\nu(x,\{x\})=f(x\mid\emptyset)=\alpha$. Proceeding inductively, then, let S_x^- and S_x^+ be the set of elements in S before and after some new element x arrives and is processed by Algorithm 1, and suppose that $\nu(s,S_x^-)\geq\alpha$ for all $s\in S_x^-$. Then, if x is rejected, we have $S_x^+=S_x^-$ and so $\nu(s,S_x^+)=\nu(s,S_x^-)\geq\alpha$ for all $s\in S_x^+$. If x is accepted, then $S_x^+=S\backslash C_x+x$ and $f(x\mid S_x^-)\geq\alpha+(1+\beta)\sum_{e\in C_x}\nu(e,S_x^-)$. Thus,

$$\nu(x,S_x^+)\geq f(x\mid S_x^+-x)\geq f(x\mid S_x^-)\geq\alpha+(1+\beta)|C_x|\alpha\geq\alpha\,,$$

where the first inequality follows from property (2) of the lemma, the second from submodularity, and the third from the induction hypothesis and the assumption that x is accepted. For any other $s\in S_x^+$, we have $\{t'\in S\backslash C_x:t'\prec s\}\subseteq\{t'\in S:t'\prec s\}$ and so by property (3) of the lemma, $\nu(s,S_x^+)\geq\nu(s,S_x^-)\geq\alpha$, as required. ◀

In our analysis we will use the following structural lemma from Chekuri et al. [7] (here, restated in our notation). This lemma applies to the execution of our algorithm StreamingLocalSearch when $S_{\text{init}}=\emptyset$, and so no element is discarded upon arrival due to $x\in S_{\text{init}}$. However, we note that the execution of our algorithm is in fact exactly the same as this algorithm executed on the pretend stream ordering introduced in Section 2 to define the incremental values ν. Specifically, in each pass of our algorithm, the set S_{init} is a feasible solution produced by the preceding pass and in the pretend stream ordering, all elements of S_{init} arrive in our pretend ordering in the same relative (pretend) order as this preceding pass. It follows that whenever $x\in S_{\text{init}}$ arrives in our pretend ordering for the present pass, we have $C_x=\emptyset$ and $\nu(x,S)=\nu(x,S_{\text{init}})\geq\alpha$ by Lemma 1 (4), since x was present in the feasible solution $S=S_{\text{init}}$ at the end of the preceding pass. Thus, each $x\in S_{\text{init}}$ will first be accepted in our pretend stream ordering, and then the rest of $X\backslash S_{\text{init}}$ is processed, exactly as in StreamingLocalSearch.

Recall that we let $\tilde{A}$ be the set of all elements that were accepted by this pass of StreamingLocalSearch (and so at some point appeared in S). For each element $x\in X$, we let S_x^- be the current set S at the moment that x arrives and S_x^+ the set after x is processed. For an element e that is accepted but later *evicted* from S, let $\chi(e)$ be the incremental value $\nu(e,S)$ of e at the moment that e was evicted.

▶ **Lemma 10** (Lemma 9 of [7]). *Let $T\in\mathcal{I}$ be a feasible solution disjoint from $\tilde{A}$, and $\tilde{S}$ be the output of the streaming algorithm. There exists a mapping $\varphi:T\to 2^{\tilde{A}}$ such that:*

1. *Every $s \in \tilde{S}$ appears in the set $\varphi(t)$ for at most p choices of $t \in T$.*
2. *Every $e \in \tilde{A} \backslash \tilde{S}$ appears in the set $\varphi(t)$ for at most $p-1$ choices of $t \in T$.*
3. *For each $t \in T$:*

$$\sum_{c \in C_t} \nu(c, S_t^-) \leq \sum_{e \in \varphi(t) \backslash \tilde{S}} \chi(e) + \sum_{s \in \varphi(t) \cap \tilde{S}} \nu(s, \tilde{S}) \,.$$

Using this charging argument, we can now prove Lemma 2 directly.

▶ **Lemma 2.** *Let $f : 2^X \to \mathbb{R}_{\geq 0}$ be a submodular function. Suppose $\tilde{S}$ is the solution produced at the end of one pass of* StreamingLocalSearch *and $\tilde{A}$ be the set of all elements accepted during this pass. Then,*

$$f(OPT \cup \tilde{A}) \leq (p + \beta p - \beta) \sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e) + (p + \beta p + 1) f(\tilde{S}) + k\alpha \,.$$

Proof. Let $R = OPT \backslash \tilde{A}$. Since $S_r^- \subseteq \tilde{A}$ for all r, submodularity of f implies that

$$\sum_{r \in R} f(r \mid S_r^-) \geq \sum_{r \in R} f(r \mid \tilde{A}) \geq f(R \cup \tilde{A}) - f(\tilde{A}) = f(OPT \cup \tilde{A}) - f(\tilde{A}) \,. \tag{8}$$

For any $r \in R$, since r was rejected upon arrival,

$$f(r \mid S_r^-) \leq (1 + \beta) \sum_{c \in C_r} \nu(c, S_r^-) + \alpha \,. \tag{9}$$

Thus, applying Lemma 10 we obtain:

$$\begin{aligned}
\sum_{r \in R} f(r \mid S_r^-) &\leq (1+\beta) \sum_{r \in R} \sum_{c \in C_r} \nu(c, S_r^-) + k\alpha && \text{(by (9) and } |R| \leq k) \\
&\leq \sum_{r \in R} (1+\beta) \left[\sum_{e \in \varphi(r) \backslash \tilde{S}} \chi(e) + \sum_{s \in \varphi(r) \cap \tilde{S}} \nu(s, \tilde{S}) \right] + k\alpha && \text{(by Lemma 10 (3))} \\
&\leq (1+\beta) \left[(p-1) \sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e) + p \sum_{s \in \tilde{S}} \nu(s, \tilde{S}) \right] + k\alpha && \text{(by Lemma 10 (1) and (2))}
\end{aligned}$$

where in the last inequality we have also used Lemma 1 (4), which implies that each $\chi(e)$ and $\nu(s, \tilde{S})$ is non-negative. Combining the above inequality with (8), we obtain

$$f(OPT \cup \tilde{A}) \leq (1+\beta) \left[(p-1) \sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e) + p \sum_{s \in \tilde{S}} \nu(s, \tilde{S}) \right] + f(\tilde{A}) + k\alpha \,. \tag{10}$$

We now bound $f(\tilde{A})$ in terms of the values $\nu(s, \tilde{S})$ and $\chi(e)$. Since $S \subseteq \tilde{A}$ at all times during the algorithm, and $\chi(e) = \nu(e, S)$ at the moment e was evicted, we have $\chi(e) \geq \nu(e, \tilde{A})$ by Lemma 1 (2). Thus,

$$f(\tilde{A}) - f(\emptyset) = \sum_{a \in \tilde{A}} \nu(a, \tilde{A}) = \sum_{s \in \tilde{S}} \nu(s, \tilde{A}) + \sum_{e \in \tilde{A} \backslash \tilde{S}} \nu(e, \tilde{A}) \leq \sum_{s \in \tilde{S}} \nu(s, \tilde{S}) + \sum_{e \in \tilde{A} \backslash \tilde{S}} \chi(e) \,, \tag{11}$$

where the first equation follows from Lemma 1 (1), and the last inequality follows from Lemma 1 (2).

Combining (10) and (11) we have:

$$f(OPT \cup \tilde{A}) \leq ((1+\beta)(p-1)+1) \sum_{e \in \tilde{A} \setminus \tilde{S}} \chi(e) + ((1+\beta)p+1) \sum_{e \in \tilde{S}} \nu(s, \tilde{S}) + f(\emptyset) + k\alpha$$
$$= (p + p\beta - \beta) \sum_{e \in \tilde{A} \setminus \tilde{S}} \chi(e) + (p + \beta p + 1) \sum_{s \in \tilde{S}} \nu(s, \tilde{S}) + f(\emptyset) + k\alpha \,. \tag{12}$$

By Lemma 1 (1), we have the following bound for the second summation in (12):

$$(p + \beta p + 1) \sum_{e \in \tilde{S}} \nu(e, \tilde{S}) + f(\emptyset) = (p + \beta p + 1)[f(\tilde{S}) - f(\emptyset)] + f(\emptyset) \leq (p + \beta p + 1) f(\tilde{S}) \,.$$

Combining this and (12) we obtain:

$$f(OPT \cup \tilde{A}) \leq (p + p\beta - \beta) \sum_{e \in \tilde{A} \setminus \tilde{S}} \chi(e) + (p + \beta p + 1) f(\tilde{S}) + k\alpha$$ ◀

B Calculations for the proof of Theorem 7

We recall that

$$\gamma_i = \gamma_{i-1}\delta_i = \frac{4p\gamma_{i-1}(\gamma_{i-1}-1)}{(\gamma_{i-1}-1+p)^2} \,.$$

Then, to see that γ_i is an increasing function of γ_{i-1} for $p \geq 1$ and $\gamma_{i-1} \geq 1$, we note that:

$$\frac{d}{d\gamma_{i-1}}\gamma_i = \frac{4p(\gamma_{i-1}-1) + 4p\gamma_{i-1}}{(\gamma_{i-1}-1+p)^2} - \frac{8p\gamma_i(\gamma_{i-1}-1)}{(\gamma_{i-1}-1+p)^3}$$

$$= \frac{4p(\gamma_{i-1}-1)(\gamma_{i-1}-1+p) + 4p\gamma_i(\gamma_{i-1}-1+p) - 8p\gamma_{i-1}(\gamma_{i-1}-1)}{(\gamma_{i-1}-1+p)^3} \geq 0$$

Where the final inequality follows from $\gamma_{i-1} \geq 1$ and $p \geq 1$.

We now verify the following inequality used at the end of Theorem 7:

$$\frac{4p\left(p+1+\frac{4p}{i-1}\right)\left(p+\frac{4p}{i-1}\right)}{\left(2p+\frac{4p}{i-1}\right)^2} \leq p+1+\frac{4p}{i} \,.$$

Rearranging both sides and placing over a common denominator gives:

$$\frac{4p\left(p+1+\frac{4p}{i-1}\right)\left(p+\frac{4p}{i-1}\right)}{\left(2p+\frac{4p}{i-1}\right)^2} = \frac{4p\left((p+1)(i-1)+4p\right)\left(p(i-1)+4p\right)}{\left(2p(i-1)+4p\right)^2}$$
$$= \frac{4p\left((p+1)(i-1)+4p\right)\left(p(i-1)+4p\right)}{\left(2p(i+1)\right)^2}$$
$$= \frac{\left((i-1)(p+1)+4p\right)(i+3)}{(i+1)^2}$$
$$= \frac{(i-1)(i+3)i(p+1)+i(i+3)4p}{i(i+1)^2}$$
$$= \frac{\left(i^2+2i-3\right)i(p+1)+(i^2+3i)4p}{i(i+1)^2}$$

and

$$\begin{aligned} p+1+\tfrac{4p}{i} &= \frac{(p+1)i+4p}{i} \\ &= \frac{i(i+1)^2(p+1)+(i+1)^2 4p}{i(i+1)^2} \\ &= \frac{\left(i^2+2i+1\right)i(p+1)+\left(i^2+2i+1\right)4p}{i(i+1)^2}. \end{aligned}$$

Then, since $p \geq 1$ and $i \geq 1$,

$$\left(p+1+\tfrac{4p}{i}\right) - \frac{4p\left(p+1+\frac{4p}{i-1}\right)\left(p+\frac{4p}{i-1}\right)}{\left(2p+\frac{4p}{i-1}\right)^2} = \frac{4i(p+1)-4(i-1)p}{i(i+1)^2} \geq 0.$$

C Additional Details for the Non-Monotone Case

C.1 Guessing the value of $f(OPT)$

Guessing the value of $f(OPT)$ is a common technique in streaming submodular function maximization. Badanidiyuru et al. [1] showed how to approximate $f(OPT)$ within a constant factor using $O(\log(k))$ space in a single pass. To avoid extra complications, we show how to guess $f(OPT)$ in two passes and refer the reader to [1] for an approximation of $f(OPT)$ on the fly. Let $\tau = \max_{e \in X} f(e)$. Using submodularity, it is easy to see that $\tau \leq f(OPT) \leq k\tau$. Consider the set

$$\Lambda = \left\{2^i \mid i \in \mathbb{Z},\ \tau \leq 2^i \leq k \cdot \tau\right\}.$$

Then there exists a value $\lambda \in \Lambda$ such that $\frac{f(OPT)}{2} \leq \lambda \leq f(OPT)$. Setting the parameter $\alpha = \varepsilon\lambda/(2k)$, we get that $\alpha \in [\varepsilon f(OPT)/4k; \varepsilon f(OPT)/2k]$. The defined range of α is sufficient for the analysis[2]. Unfortunately, it is still not possible to know which $\lambda \in \Lambda$ satisfies the property. However, it suffices to run the randomized local-search algorithm for every $\lambda \in \Lambda$ in parallel and output the best solution of all the copies. This operation increases the space complexity by a multiplicative $O(\log_2 k)$ factor, and adds one additional pass to find τ.

C.2 Proof of Theorem 12

Here we give a full proof of the following theorem from Section 4:

▶ **Theorem 9.** *Let $\mathcal{M}^p = (X, \mathcal{I})$ be a p-matchoid of rank k and let $f\colon 2^X \to \mathbb{R}_{\geq 0}$ be a non-negative submodular function. Suppose there exists an algorithm for the offline instance of the problem with approximation factor $\bar{\gamma}_{\text{off}}$. For any $\varepsilon > 0$, the randomized streaming local-search algorithm returns a solution $\bar{S} \in \mathcal{I}$ such that*

$$f(OPT) \leq (p+1+\bar{\gamma}_{\text{off}}+O(\varepsilon))\,\mathbb{E}[f(\bar{S})]$$

using a total space of $O\left(\frac{p^3 k \log_2 k}{\varepsilon^3}\right)$ and $O\left(\frac{p}{\varepsilon}\right)$-passes.

[2] Equation (6) and the bound $\alpha k \leq \varepsilon f(OPT)$ are where we need the exact value of α, using upper and lower bounds for α yield the same result up to the hidden constant in the term $O(\epsilon)$.

In the same spirit as in Section 3, we show that we can derive a guarantee with respect to the solution $\mathbb{E}[f(S_i)]$ produced after the i^{th} pass even when the function is non-monotone. In fact, we show that the analysis of the non-monotone case reduces to the monotone case as shown in the following theorem.

▶ **Theorem 11.** *Let f be a non-negative submodular function. Assume $\alpha = \varepsilon f(OPT)/2k$ and let $d \geq i > 1$. Suppose that at the start of the i^{th} iteration of the randomized local-search algorithm with a buffer of size $m = 4dk/\varepsilon^2$ we have $(1-\varepsilon)f(OPT) \leq \gamma_{i-1}\,\mathbb{E}[f(S_{i-1})] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_{i-1})]$. Then,*

$$(1-\varepsilon)f(OPT) \leq \min\left\{\gamma_{i-1}\delta_i, \left(\frac{p}{\beta_i} + p - 1\right)(1-\delta_i) + p + \beta_i p + 1\right\} \cdot \mathbb{E}[f(S_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)],$$

where $\delta_i = \frac{\mathbb{E}[f(S_{i-1})]}{\mathbb{E}[f(S_i)]}$.

Proof. From the definition of γ_{i-1} and δ_i, it follows that,

$$(1-\varepsilon)f(OPT) \leq \gamma_{i-1}\,\mathbb{E}[f(S_{i-1})] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_{i-1})] \leq \gamma_{i-1}\delta_i\,\mathbb{E}[f(S_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)]\,. \quad (13)$$

where in the last inequality we have used the definition of δ_i and the fact that $f(S'_i) \geq f(S'_{i-1})$, which follows from the way S'_i is defined in Algorithm 3.

On the other hand, $\mathbb{E}[f(S_i)] - \mathbb{E}[f(S_{i-1})] = (1-\delta_i)\,\mathbb{E}[f(S_i)]$. Thus, by (7) we also have:

$$\begin{aligned}
&(1-\varepsilon)f(OPT)\\
&\quad\leq \left(\tfrac{p}{\beta_i} + p - 1\right)(\mathbb{E}[f(S_i)] - \mathbb{E}[f(S_{i-1})]) + (p + \beta p + 1)\,\mathbb{E}[f(S_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)]\\
&\quad= \left(\left(\tfrac{p}{\beta_i} + p - 1\right)(1-\delta_i) + p + \beta_i p + 1\right)\mathbb{E}[f(S_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)]\,. \qquad (14)
\end{aligned}$$

Since the right-hand side of equation 13 is an increasing function of δ_i and the right-hand side of equation 14 is a decreasing function of δ_i, the guarantee we obtain is always at least as good as that obtained when these two values are equal. ◀

As in the monotone case, the lemma enables us to derive values of β so as to minimize the value of the approximation ratio. The following follows directly from the same calculations as in Section 3 and Appendix B.

▶ **Theorem 12.** *Suppose we run Algorithm 3 with a buffer of size $m = 4dk/\varepsilon^2$ on a arbitrary p-matchoid constraint and a submodular function, with $\alpha = \varepsilon f(OPT)/2k$, $\beta_1 = 1$ and $\beta_i = \frac{\gamma_{i-1}-1-p}{\gamma_{i-1}-1+p}$ where γ_i is given by the recurrence, $\gamma_1 = 4p$ and $\gamma_i = \frac{4p\gamma_{i-1}(\gamma_{i-1}-1)}{(\gamma_{i-1}-1+p)^2}$. Then,*

$$(1-\varepsilon)f(OPT) \leq \left(p + 1 + \frac{4p}{i}\right)\mathbb{E}[f(\tilde{S}_i)] + \bar{\gamma}_{\text{off}}\,\mathbb{E}[f(S'_i)].$$

In particular after $d = \frac{4p}{\varepsilon}$ passes,

$$(1-\varepsilon)f(OPT) \leq (p + 1 + \bar{\gamma}_{\text{off}} + \varepsilon)\,\mathbb{E}[f(\bar{S}_d)]\,.$$

Under a matroid constraint, Algorithm 3 with $\alpha = \varepsilon f(OPT)/2k$, $\beta_i = 1/i$ and $d = 2\varepsilon^{-1}$ passes outputs a solution $\bar{S}$ such that,

$$(1-\varepsilon)f(OPT) \leq (2 + \bar{\gamma}_{\text{off}} + \varepsilon)\,\mathbb{E}[f(\bar{S})]\,,$$

where $\bar{\gamma}_{\text{off}}$ is the approximation ration of the best offline algorithm for maximizing f under a matroid constraint.

We are now ready to complete the proof of Theorem 9. We assume that we know the value of $f(OPT)$ before hand, which can be accomplished approximately as in Section C.1. Let $\varepsilon' = \varepsilon/p$ with $1/2 \geq \varepsilon' > 0$ and let $\alpha = \varepsilon' f(OPT)/2k$. We want to obtain an additive error term instead of a multiplicative error term as stated in Theorem 12. By Theorem 12,

$$(1-\varepsilon')f(OPT) \leq \left(p+1+\bar{\gamma}_{\text{off}}+\frac{4p}{d}\right)\mathbb{E}[f(\bar{S}_d)] = (p+1+\bar{\gamma}_{\text{off}})\left(1+O\left(d^{-1}\right)\right)\mathbb{E}[f(\bar{S}_d)]\,.$$

Using the fact that $(1-\varepsilon')^{-1} \leq 1+2\varepsilon'$ for $\varepsilon' \in (0, 1/2]$, we get that,

$$f(OPT) \leq (p+1+\bar{\gamma}_{\text{off}})\left(1+O\left(d^{-1}\right)\right)(1+2\varepsilon')\,\mathbb{E}[f(\bar{S}_d)]\,. \tag{15}$$

Since $\varepsilon' = \varepsilon/p$, setting $d = O(p/\varepsilon)$ we finally obtain the desired result:

$$f(OPT) \leq (p+1+\bar{\gamma}_{\text{off}})(1+O(\varepsilon/p))(1+2\varepsilon/p)\,\mathbb{E}[f(\bar{S}_d)] \leq (p+1+\bar{\gamma}_{\text{off}}+O(\varepsilon))\,\mathbb{E}[f(\bar{S}_d)].$$

For the space complexity, we note that the randomized local-search algorithm stores the buffer B and maintains two past solutions $S_i, S_i' \in \mathcal{I}$, together with the current solution $S \in \mathcal{I}$. Hence, the total space needed is equal to $O(|B| + |S_i'| + |S_i| + |S|) = O(m+3k) = O\left(p^3 k \varepsilon^{-3}\right)$, times an additional factor of $O(\log k)$ for guessing $f(OPT)$. The number of passes is $d = O(p/\varepsilon)$.

Polylogarithmic Approximation Algorithm for k-Connected Directed Steiner Tree on Quasi-Bipartite Graphs

Chun-Hsiang Chan
Department of Computer Science, University of Michigan, Ann Arbor, MI, USA
kenhchan@umich.edu

Bundit Laekhanukit
Institute for Theoretical Computer Science, Shanghai University of Finance & Economics, China
http://itcs.shufe.edu.cn/~blaekh
bundit@sufe.edu.cn

Hao-Ting Wei
Department of IEOR, Columbia University, New York, NY, USA
hw2738@columbia.edu

Yuhao Zhang
Department of Computer Science, The University of Hong Kong, China
https://i.cs.hku.hk/~yhzhang2/
yhzhang2@cs.hku.hk

Abstract

In the k-Connected Directed Steiner Tree problem (k-DST), we are given a directed graph $G = (V, E)$ with edge (or vertex) costs, a root vertex r, a set of q terminals T, and a connectivity requirement $k > 0$; the goal is to find a minimum-cost subgraph H of G such that H has k edge-disjoint paths from the root r to each terminal in T. The k-DST problem is a natural generalization of the classical Directed Steiner Tree problem (DST) in the fault-tolerant setting in which the solution subgraph is required to have an r, t-path, for every terminal t, even after removing $k - 1$ vertices or edges. Despite being a classical problem, there are not many positive results on the problem, especially for the case $k \geq 3$. In this paper, we present an $O(\log k \log q)$-approximation algorithm for k-DST when an input graph is quasi-bipartite, i.e., when there is no edge joining two non-terminal vertices. To the best of our knowledge, our algorithm is the only known non-trivial approximation algorithm for k-DST, for $k \geq 3$, that runs in polynomial-time Our algorithm is tight for every constant k, due to the hardness result inherited from the Set Cover problem.

2012 ACM Subject Classification Theory of computation → Routing and network design problems

Keywords and phrases Approximation Algorithms, Network Design, Directed Graphs

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.63

Category APPROX

Related Version The arXiv version is available at https://arxiv.org/abs/1911.09150.

Funding *Bundit Laekhanukit*: Partially supported by Science and Technology Innovation 2030 – "New Generation of Artificial Intelligence" Major Project No.(2018AAA0100903), NSFC grant 61932002, Program for Innovative Research Team of Shanghai University of Finance and Economics (IRTSHUFE) and the Fundamental Research Funds for the Central Universities. Also, partially supported by the 1000-talent award by the Chinese Government.

Acknowledgements The works were initiated while all the authors were at the Institute for Theoretical Computer Science at the Shanghai University of Finance and Economics, and the work were done while Chun-Hsiang Chan and Hao-Ting Wei were in bachelor and master programs in Computer Science at the Institute of Information Science, Academia Sinica, Taipei.



© Chun-Hsiang Chan, Bundit Laekhanukit, Hao-Ting Wei, and Yuhao Zhang;
licensed under Creative Commons License CC-BY



1 Introduction

Designing a network that can operate under failure conditions is an important task for Computer Networking in both theory and practice. Many models have been proposed to capture this problem, giving rise to the area of survivable and fault-tolerant network design. In the past few decades, there have been intensive studies on the survivable network design problems; see, e.g., [55, 30, 35, 20, 14, 46, 32]. The case of link-failure is modeled by the *Edge-Connectivity Survivable Network Design* problem (EC-SNDP), which is shown to admit a 2-approximation algorithm by Jain [35]. The case of node-failure is modeled by the *Vertex-Connectivity Survivable Network Design* problem (VC-SNDP), which is shown to admit a polylogarithmic approximation algorithm by Chuzhoy and Khanna [14]. Nevertheless, most of the known algorithmic results pertain to only undirected graphs, where each link has no prespecified direction. In the directed case, only a few results are known as the general case of Survivable Network Design is at least as hard as the *Label-Cover* problem [16], which is believed to admit no sub-polynomial approximation algorithm [44, 2].

This paper studies the special case of the Survivable Network Design problem on directed graphs, namely the *k-Connected Directed Steiner Tree* problem (k-DST), which is also known as the *Directed Root k-Connectivity*. In this problem, we are given an n-vertex directed graph $G = (V, E)$ with edge-costs $c : E \to \mathbb{R}_0^+$, a root vertex r, a set of q terminals $T \subseteq V - \{r\}$ and a connectivity requirement $k \in \mathbb{Z}^+$; the goal is to find a minimum-cost subgraph $H \subseteq G$ that has k edge-disjoint[1] r, t-paths for every terminal $t \in T$. This problem was mentioned in [19] and have been subsequently studied in [12, 41, 8, 43, 32]. The only known non-trivial approximation algorithms for k-DST is for the case $k = 2$ due to the work of Grandoni and Laekhanukit [32], and for the case of γ-shallow instances due to the work of Laekhanukit [43]. To the best of our knowledge, for $k \geq 3$, there were only a couple of positive results on k-DST: (1) Laekhanukit [43] devised an approximation algorithm whose running time and approximation ratios depend on the diameter of the optimal solution, and (2) Chalermsook, Grandoni and Laekhanukit [8] devised a bi-criteria approximation algorithm for a special case of k-DST, namely the *k-Edge-Connected Group Steiner Tree* (k-GST), where the solution subgraph is guaranteed to be an $O(\log^2 n \log k)$-approximate solution, whereas the connectivity is only guaranteed to be at least $\Omega(k/\log n)$. We focus on the case of k-DST where an input graph is *quasi-bipartite*, i.e., there is no edge joining any pair of non-terminal (Steiner) vertices, which generalizes the works of Hibi and Fujito [34], and Friggstad, Könemann and Shadravan [26] for the classical *directed Steiner tree* problem (the case $k = 1$).

The main contribution of this paper is an $O(\log q \log k)$-approximation algorithm for k-DST on quasi-bipartite graphs, which runs in polynomial-time regardless of the structure of the optimal solution. Our result can be considered the first true polylogarithmic approximation algorithm whose running time is independent of the structure (i.e., diameter) of the optimal solution, albeit the algorithm is restricted to the class of quasi-bipartite graphs. Our technique is different from all the previous works on k-DST [32, 43, 8]; all these results rely on the tree-rounding algorithm for the *Group Steiner Tree* problem by Garg, Konjevod and Ravi [27], and thus require either an LP whose support is a tree or a tree-embedding technique (e.g., Räcke's decomposition [50] as used in [8]). Our algorithm, on the other hand, employs the *Halo-Set decomposition* devised by Kortsarz and Nutov [39] and further developed in a series

[1] While we define the problem here as an edge-connectivity problem, our algorithm itself works for both edge and vertex connectivity variants, and can handle both edge and vertex costs.

of works [17, 11, 46, 48, 42, 47]. It is worth noting that the families of subsets decomposed from our algorithm are not uncrossable. We circumvent this difficulty by reducing it to the *Set Cover* problem. Our algorithm can be seen as a variant of the *spider decomposition* method developed by Klein and Ravi [36], and Nutov [45].

Lastly, we remark that it was discussed in [32] that the tree-embedding approach reaches the barrier as soon as $k > 2$, and this holds even for quasi-bipartite graphs. Please see Appendix B for discussions. While our algorithm exploits the structure of quasi-bipartite graphs, we hope that our technique using the Halo-Set decomposition would be an alternative method that sheds some light on developing approximation algorithms for the general case of k-DST for $k > 2$.

1.1 Related Works

Directed Steiner tree has been a central problem in combinatorial optimization. There have been a series of work studying this problem; see, e.g., [56, 9, 54, 25, 33, 29]. The best approximation ratio of $O(q^\epsilon)$, for any $\epsilon < 0$, in the regime of polynomial-time algorithms, is known in the early work of Charikar et al. [9][2], which leads to an $O(\log^3 q)$-approximation algorithm that runs in quasi-polynomial-time. Very recently, Grandoni, Laekhanukit and Li [33] developed a framework that gives a quasi-polynomial-time $O(\log^2 q/\log\log q)$-approximation algorithm for the Directed Steiner Tree problem, and this approximation ratio is the best possible for quasi-polynomial-time algorithms, assuming the *Projection Games Conjecture* and $\text{NP} \subseteq \bigcup_{\delta>0} \text{ZPTIME}(2^{n^\delta})$. The same approximation ratio was obtained in an independent work of Ghuge and Nagarajan [29].

The study of Steiner tree problems on quasi-bipartite graphs was initiated by Rajagopalan and Vazirani [51] in order to understand the bidirected-cut relaxation of the (undirected) *Steiner tree* problem. Since then the special case of quasi-bipartite graphs has played a central role in studying the Steiner tree problem; see, e.g., [52, 6, 53, 37, 5, 31]. For the case of directed graphs, Hibi and Fujito [34], Friggstad, Könemann and Shadravan [26] independently discovered $O(\log n)$-approximation algorithms for the directed Steiner tree problem on quasi-bipartite graphs. Assuming $\text{P} \neq \text{NP}$, this matches to the lower bound of $(1-\epsilon)\ln n$, for any $\epsilon > 0$, inherited from the Set Cover problem [18, 15].

The generalization of the Steiner tree problem is known as the Survivable Network Design problem, which has been studied in both edge-connectivity [55, 30, 35], vertex-connectivity [14] and element-connectivity [20] settings. The edge and element connectivity Survivable Network Design problems admit factor 2 approximation algorithms via the iterative rounding method, while the vertex-connectivity variant admits no polylogarithmic approximation algorithm [38, 7, 41] unless $\text{NP} \subseteq \text{DTIME}(n^{\text{polylog}(n)})$. To date, the best approximation ratio known for the Vertex-Connectivity Survivable Network problem is $O(k^3 \log n)$ due to the work of Chuzhoy and Khanna [14]. The single-source k-vertex-connectivity variant, which is closely related to the problem considered in this paper, has been studied in [7, 13, 10, 46, 49], culminating in the best approximation ratio of $O(k \log k)$ due to Nutov [46].

In vertex-connectivity network design, one of the most common techniques is the Halo-Set decomposition method, which has been developed in a series of works [39, 17, 11, 48]. The main idea is to use the number of minimal deficient sets as a notion of progress. Here a deficient set is a subset of vertices that needs at least one incoming edge to satisfy the connectivity requirement. The minimal deficient sets in [39, 17, 11, 48], called cores, are

[2] The same result can be obtained by applying the algorithm by Kortsarz and Peleg in [40]

independent and have only polynomial number, while the total number of deficient sets is exponential on the number of vertices. The families of deficient sets defined by these cores allow us to keep track of how many deficient sets remain in a solution subgraph. The early version of this method can be traced back to the seminal result of Frank [22] and that of Frank and Jordan [23]; please see [24] for reference therein.

The spider decomposition method was introduced by Klein and Ravi [36] to handle the *Vertex-Weighted Steiner Tree* problem. This technique gives a tight approximation result (up to constant factor) to the problem. Later, Nutov generalized the technique to deal with the *Minimum Power-Cover* problems [45] and subsequently for the *Vertex-Weighted Element-Connectivity Survivable Network Design* problem [46].

1.2 Our Result

The main result in our paper is an $O(\log q \log k)$-approximation algorithm for k-DST on quasi-bipartite graphs. To keep the flow, our algorithm is presented as a randomized algorithm. The derandomization is provided in Appendix A. Since our algorithm is LP-based, it also gives an upper bound on the integrality gap of the standard LP-relaxation.

▶ **Theorem 1.** *Consider the k-Connected Directed Steiner Tree problem where an input graph consists of an n-vertex quasi-bipartite graph and a set of q terminals. There exists a polynomial-time $O(\log q \log k)$-approximation algorithm. Moreover, the algorithm gives an upper bound on the integrality gap of $O(\log q \log k)$ for the standard cut-based LP-relaxation of the problem.*

2 Preliminaries

We use standard graph terminologies. Given a graph G, we denote by $V(G)$ and $E(G)$ the vertex set and edge set of G, respectively. For any subset of vertices $U \subseteq V(G)$, we denote by $\delta^{in}_G(U)$ the set of edges in G entering the set U and denote by $\deg^{in}_G(U)$ its cardinality. We denote by $E_G(U)$ the set of edges that have both head and tail in U. That is,

$$\delta^{in}_G(U) = \{wv \in E(G) : v \in U, w \notin U\}, \quad \deg^{in}_G(U) = |\delta^{in}_G(U)|, \quad \text{and}$$
$$E_G(U) = \{vw \in E(G) : v, w \in U\}.$$

We will omit the subscript G if the graph G is known in the context, and we may replace E_G with another edge-set, e.g., E_+. Then, for any subset of edges E', we denote the total cost of edges in E' by $\mathsf{cost}(E') = \sum_{e \in E'} c_e$.

2.1 Problem Definitions

k-Edge-Connected Directed Steiner Tree (k-DST)

In the *k-Edge-Connected Directed Steiner Tree* problem (k-DST), we are given a graph G with non-negative edge-costs $c : E \longrightarrow \mathbb{R}^+_0$, a root vertex r and a set of q terminals $T \subseteq (V(G) - \{r\})$, and the goal is to find a minimum-cost subgraph $H \subseteq G$ such that H has k edge-disjoint $r \to t$-paths for every terminal $t \in T$.

Rooted Connectivity Augmentation (Rooted-Aug)

In Rooted-Aug, we are given a graph G with the edge-set $E(G) = E_0 \cup E_+$, where E_0 is the set of zero-cost edges and E_+ is the set of positive-cost edges, a root vertex r and a set of terminals $T \subseteq V(G) - r$ such that E_0 induces a subgraph $G_0 \subseteq G$ that has ℓ edge-disjoint

$r \to t$-paths for every terminal $t \in T$. The goal in this problem is to find a minimum-cost subset of edges $E' \subseteq E_+$ such that $E_0 \cup E'$ induces a subgraph $H \subseteq G$ that has $\ell + 1$ edge-disjoint $r \to t$-paths for every terminal $t \in T$.

We may phrase Rooted-Aug as a problem of covering deficient sets as follows. We say that a subset of vertices $U \subseteq V(G)$ is a *deficient set* if U separates the root vertex r and some terminal $t \in T$, but U has less than $\ell + 1$ incoming edges (which means that U has exactly ℓ incoming edges); that is, U is a deficient set if $r \notin U$, $U \cap T \neq \emptyset$ and $\deg^{in}_{G_0}(U) = \ell$. These subsets of vertices need at least one incoming edge to satisfy the connectivity requirement. We say that an edge $e \in E_+$ *covers* a deficient set U if $\deg^{in}_{E_0 \cup \{e\}}(U) \geq \ell$, which means that adding e to G_0 satisfies the connectivity requirement on U.

Let $\mathcal{F}$ denote the set of all deficient sets in the graph G_0. Then Rooted-Aug may be phrased as the problem of finding a minimum-cost subset of edges $E' \subseteq E_+$ that covers all the deficient sets, which can be described by the following optimization problem:

$$\min\{E' \subseteq E_+ : \deg^{in}_{E'}(U) \geq 1 \ \forall U \in \mathcal{F}\}.$$

Set Cover

Given a universe $\mathcal{U}$ of n elements and a collection of m subsets $S_1, \ldots, S_m \subseteq \mathcal{U}$, each associated with weight w_j, for $j = 1, \ldots, m$, the goal in the Set Cover problem is to find a collection $\mathcal{S}^*$ of subsets with minimum total weights so that the union of all subsets in $\mathcal{S}^*$ is equal to $\mathcal{U}$.

2.2 Deficient Sets, Cores and Halo-families

This section discusses subsets of vertices called deficient sets that certify that the current solution subgraph in Rooted-Aug (and also in k-DST) does not meet the connectivity requirement. To be formal, a subset of vertices $U \subseteq V(G)$ is called a *deficient set* in the graph G if $T \cap U \neq \emptyset$, $r \notin U$ and $\deg^{in}_G(U) < k$; that is, $(V(G) - U, U)$ induces an edge-cut of size $< k$ that separates some terminal $t \in U \cap T$ from the root vertex r. We say that an edge $vw \notin E(G)$ *covers* a deficient set U if $\deg^{in}_{G+vw}(U) \geq k$, i.e., the set U is not a deficient set after adding the edge vw. Similarly, we say that a subset of edges E' covers a deficient set or a collection of deficient sets $\mathcal{F}$ if $\deg^{in}_{G+E'}(U) \geq k$, for every deficient set $U \in \mathcal{F}$.

Let $\mathcal{F}$ be a family of deficient sets. A *core* $C \in \mathcal{F}$ is a deficient set such that there is no deficient set in $\mathcal{F}$ properly contained in C. The *Halo-family* $\mathsf{Halo}(C)$ of a core C is the collection of all deficient sets in $\mathcal{F}$ that contain C but no other core $C' \neq C$. The *Halo-set* $H(C)$ of C is the union of all the deficient sets in $\mathsf{Halo}(C)$, i.e., $H(C) = \bigcup_{U \in \mathsf{Halo}(C)} U$.

2.3 LP-relaxations

Throughout this paper, we will use the following standard (cut-based) LP-relaxation for k-DST and Rooted-Aug. Our LP-relaxations will be written in terms of deficient sets. We denote by $\mathsf{Val}(z)$ the cost of the optimal solution to an LP z.

LP for k-DST

Here we present the standard cut-based LP-relaxation for k-DST, denoted by $\mathsf{LP}(k)$. The collection of deficient sets in this LP is defined by $\mathcal{F}(k) = \{U \subseteq V - \{r\} : U \cap T \neq \emptyset\}$.

$$\mathsf{LP}(k) = \begin{cases} \min & \sum_{e \in E} c_e x_e & \\ \text{s.t.} & & \\ & \sum_{e \in \delta^{in}_G(U)} x_e \geq k & \forall U \in \mathcal{F}(k) \\ & 0 \leq x_e \leq 1 & \forall e \in E(G) \end{cases}$$

LP for Rooted-Connectivity Augmentation

Here we assume that the initial graph G_0 is already ℓ-rooted-connected, and the goal is to add edges to increase the connectivity of the solution subgraph by one. Thus, the collection of deficient sets in this problem is defined by $\mathcal{F}(\ell) = \{U \subseteq V : U \cap T \neq \emptyset, \deg^{in}_{G_0}(U)| = \ell\}$. Below is the standard cut-based LP-relaxation for the problem of increasing the rooted-connectivity of a graph by one.

$$\mathsf{LP}^{aug}(\ell) = \begin{cases} \min & \sum_{E(G)-E(G_0)} c_e x_e \\ \text{s.t.} & \\ & \sum_{e \in \delta^{in}_{E(G)-E(G_0)}(U)} x_e \geq 1 \quad \forall U \in \mathcal{F}(\ell) \\ & 0 \leq x_e \leq 1 \qquad\qquad\quad \forall e \in E(G) - E(G_0) \end{cases}$$

3 Properties of Deficient Sets in Rooted Connectivity Augmentation

This section presents the basic properties of deficient sets, cores and Halo-families in a Rooted-Aug instance, which will be used in the analysis of our algorithm. Readers who are familiar with these properties may skip this section. Similar lemmas and proofs can be seen, e.g., in [11]. Our proofs are rather standard. The readers who are familiar with these properties may skip to the next section.

The first property is the uncrossing lemma for deficient sets of Rooted-Aug.

▶ **Lemma 2** (Uncrossing Properties). *Consider an instance of Rooted-Aug. Let G_0 be a rooted ℓ-connected graph, and let A, B be deficient sets in G_0 that have a common terminal, i.e., $A \cap B \cap T \neq \emptyset$. Then both $A \cup B$ and $A \cap B$ are deficient sets.*

Proof. We prove the lemma by using Menger's theorem and the submodularity of the indegree function $\deg^{in}$. First, since G_0 is rooted ℓ-connected, we know from Menger's Theorem that $\deg^{in}(A)$ and $\deg^{in}(B) = \ell$. We also know that $\deg^{in}(A \cup B) \geq \ell$ and $\deg^{in}(A \cap B) \geq \ell$ because the root r is not contained in either A or B and that $A \cap B \cap T \neq \emptyset$. By the submodularity of $\deg^{in}$, it holds that

$$2\ell = \deg^{in}(A) + \deg^{in}(B) \geq \deg^{in}(A \cup B) + \deg^{in}(A \cap B) \geq 2\ell.$$

Therefore, $\deg^{in}(A \cup B) = \deg^{in}(A \cap B) = \ell$, implying that both $A \cup B$ and $A \cap B$ are deficient sets in the Rooted-Aug instance. ◀

The next lemma gives an important property of the cores arose from deficient sets in directed graphs; that is, two cores may have non-empty intersection on Steiner vertices, but they are disjoint on terminal vertices.

▶ **Lemma 3** (Members of Two Halo-families are Terminal Disjoint). *Let C and C' be two distinct cores. Then, for any deficient sets $U \in \mathsf{Halo}(C)$ and $U' \in \mathsf{Halo}(C)$, it holds that $U \cap U' \cap T = \emptyset$, i.e., any members of two distinct Halo-families have no common terminals.*

Proof. We prove the lemma by contradiction. Let U and U' be deficient sets $U \in \mathsf{Halo}(C)$ and $U' \in \mathsf{Halo}(C)$ such that U and U' share a terminal $t \in U \cap U' \cap T$. We may assume that U and U' are minimal such sets, i.e., there are no deficient sets $W \in \mathsf{Halo}(C)$ and $W' \in \mathsf{Halo}(C)$ such that (1) W is properly contained in U, (2) W' is properly contained in U' and (3) $t \in W \cap W'$. By Lemma 2, $U \cap U'$ must be a deficient set properly contained in both U and U' (because $C \neq C'$). This contradicts the minimality of U and U'. ◀

Then we combine Lemma 3 and that there is no edge joining any pair of Steiner vertices in quasi-bipartite graphs, we have the Internally Edge-Disjoint Lemma.

▶ **Lemma 4** (Internally Edge-Disjoint). *Consider a quasi-bipartite graph G. For any edge $e \in E(G)$, there is at most one core $C \in \mathcal{C}$ such that $e \in E(H(C))$.*

Proof. Consider any edge $uv \in E(G)$. Since G is a quasi-bipartite graph, one of u and v must be a terminal. By Lemma 3, we know that there can be at most one Halo-family $\mathsf{Halo}(C)$, for some $C \in \mathcal{C}$, whose member contains both u and v. Hence, the lemma follows. ◀

The next lemma shows that both the union and the intersection of any two deficient sets in a Halo-family $\mathsf{Halo}(C)$ are also deficient sets in $\mathsf{Halo}(C)$. This is a crucial property for computing the halo-set $H(C)$ as we are unable to list all the deficient sets in a Halo-family.

▶ **Lemma 5** (Union and Intersection of Halo-Family Members). *Let $\mathcal{F}$ be a family of all deficient sets in G_0, and let C be any core w.r.t. $\mathcal{F}$. Then, for any two deficient sets $A, B \in \mathsf{Halo}(C)$, both $A \cap B$ and $A \cup B$ are also deficient sets in $\mathsf{Halo}(C)$.*

Proof. Consider any deficient sets $A, B \in \mathsf{Halo}(C)$. Since both A and B contain C, they share at least one terminal. Thus, Lemma 2 implies that both $A \cup B$ and $A \cap B$ are deficient sets. Clearly, $A \cap B$ contains C and no other core $C' \neq C$. Thus, $A \cap B$ is a member of $\mathsf{Halo}(C)$.

Next consider $A \cup B$. Assume for a contradiction that $A \cup B$ is not a member of $\mathsf{Halo}(C)$. Then $A \cup B$ must contain a core $C' \neq C$. This means that at least one of the sets, say A, contains some terminal $t \in C'$. By Lemma 2, since A and C' have a common terminal, it holds that $A \cap C'$ is a deficient set. Since $C' \subsetneq A$ (because A is a member of $\mathsf{Halo}(C)$), we have that $A \cap C'$ is a deficient set that is strictly contained in C', a contradiction. ◀

It follows as a corollary that $H(C) = \bigcup_{U \in \mathsf{Halo}(C)} U$ is a also deficient set in $\mathsf{Halo}(C)$.

▶ **Corollary 6** (Halo-set is deficient). *Let $\mathcal{F}$ be a family of all deficient sets in G_0, and let C be any core w.r.t. $\mathcal{F}$. Then the Halo-set $H(C) = \bigcup_{U \in \mathsf{Halo}(C)} U$ is also a deficient set in $\mathsf{Halo}(C)$.*

Corollary 6 implies that $H(C)$ can be computed in polynomial-time using an efficient maximum-flow algorithm. Such an algorithm can be seen in [11].

▶ **Corollary 7.** *For any core C, its Halo-set $H(C) = \bigcup_{U \in \mathsf{Halo}(\mathcal{C})} U$ can be computed in polynomial-time.*

4 Our Algorithm and Its Overview

This section provides the overview of our algorithm, which is based on the *connectivity augmentation* framework plus the Halo-set decomposition method. To be specific, our algorithm starts with an empty graph called $H_0 = (V, \emptyset)$. Then we add edges from G to the graph H_0 to form a graph H_1 that has at least one path from the root vertex r to each terminal $t \in T$. We keep repeating the process, which produces graphs $H_2, \ldots, H_k$ such that each graph H_ℓ, for $\ell \in [k]$, has ℓ edge-disjoint r,t-paths for every terminal $t \in S$. In each iteration $\ell \in [k]$, we increase the rooted-connectivity of a graph by one using the Halo-set decomposition method.

We discuss the connectivity augmentation framework in Section 4.1 and discuss the algorithm based on the Halo-set decomposition method for Rooted-Aug in Section 4.2. We devote Section 5 to present a key subroutine for solving the the problem of *covering Halo-families* via a reduction to the Set Cover problem.

4.1 Connectivity Augmentation Framework

A straightforward analysis of the connectivity augmentation framework incurs a factor k in the approximation ratio. Nevertheless, provided that the approximation algorithm for Rooted-Aug is based on the standard LP for k-DST, the cost incurred by this framework is only $\sum_{\ell=1}^{k} 1/(k-\ell+1) = O(\log k)$. This is known as the *LP-scaling technique*, which has been used many times in literature; see, e.g., [30, 39, 12].

▶ **Lemma 8** (LP-Scaling). *Consider an instance of the k-DST problem, and its corresponding LP, namely $\mathsf{LP}(k)$. Suppose there exists an algorithm that produces an integer solution to $\mathsf{LP}^{aug}(\ell)$ with costs at most $\alpha_\ell \cdot \mathsf{Val}(\mathsf{LP}^{aug}(\ell))$. Then there exists an $\sum_{\ell=1}^{k} \alpha_\ell/(k-\ell+1) = O(\alpha \log k)$ approximation algorithm for k-DST, where $\alpha = \max_{\ell=1}^{k} \alpha_\ell$.*

Proof. Let G be the input graph in the k-DST instance. Let H^* be an optimal integral solution to k-DST (and thus $\mathsf{LP}(k)$), and let $G_0 \subseteq G$ be the initial solution subgraph of Rooted-Aug where we wish to increase the connectivity of G_0 from ℓ to $\ell+1$ by adding edges from $E(G) - E(G_0)$. Then we can define the following LP solution $\{x_e\}_{e \in E(G)}$ to $\mathsf{LP}^{aug}(\ell)$:

$$x_e = \begin{cases} \frac{1}{k-\ell} & \text{if } e \in E(H^*) - E(G_0) \\ 0 & \text{otherwise.} \end{cases}$$

Let $\mathcal{F}$ be the family of deficient sets in the Rooted-Aug instance. Then we know by Menger's theorem that any deficient set $U \in \mathcal{F}$ has at least k incoming edges in H^*, and at most ℓ of them are in G_0 (because $\deg^{in}_{G_0}(U) = \ell$ by the definition of the deficient set). Consequently, we have

$$\sum_{e \in \delta^{in}_{E(G)-E(G_0)}(U)} x_e \geq (k-\ell) \cdot \frac{1}{k-\ell} = 1.$$

This means that $\{x_e\}_{e \in E(G)-E(G_0)}$ is a feasible solution to $\mathsf{LP}^{aug}(\ell)$ whose cost is at most $(1/(k-\ell))\mathsf{Val}(\mathsf{LP}(k))$. The lemma then follows by taking the summation over all $\ell = 0, 1, \ldots, k-1$. ◀

4.2 Algorithm for Rooted-Aug via Halo-set Decomposition

The algorithm for rooted-connectivity augmentation is built on the Halo-set Decomposition framework. In detail, we decompose vertices in the graph G_0 into a collection of subsets of vertices, each is defined by a Halo-family $\mathsf{Halo}(C)$, which is in turn defined by its core C. Then we add edges to cover all the deficients that are contained in any of these families. However, the collection of Halo-families does not include all the deficient sets in the graph because a deficient set that contain two distinct cores are not recorgnized by any Halo-families. Thus, after we cover all these Halo-families (i.e., we add edges covering all its members), we need to recompute the deficient sets remaining in the graph and form the system of Halo-families again.

Following the above method, our algorithm runs in multiple iterations. In each iteration, we first compute all the cores and their corresponding Halo-set in the current solution subgraph, which can be done in polynomial time. (We recall that it is not possible to compute a Halo-family explicitly because it may contain exponential number of deficient sets.) These cores define a collection of Halo-families. Our goal is then to find a subset of edges E' that covers Halo-families in this collection. To be formal, by *covering a Halo-family*, we mean that we find a subset of edges that covers every deficient set in its family. Here

our algorithm departs from the previous application of the Halo-set decomposition as we are not aiming to cover all the Halo-families. We cover only a constant fraction of Halo-families from the collection, which is sufficient for our purposes. Once we found the subset of edges E', we add it to the solution subgraph and recompute the cores and their Halo-sets.

To find a set of edges E', we need to compute an optimal solution to the LP for augmenting the connectivity of a graph from ℓ to $\ell+1$ (i.e., $\mathsf{LP}^{aug}(\ell)$), denoted by $\{x_e\}_{e\in E_+}$, where E_+ is the set of edges not in the initial solution subgraph H_ℓ, which is ℓ-rooted-connected. Using this LP-solution, we can find a set of edges E' that covers at least $1/9$ fraction of the collection of Halo-families whose cost is at most $4\sum_{e\in E_+} c_e x_e$ via a reduction to the Set Cover problem. This subroutine is presented in Section 5. Note that the mentioned subroutine is a randomized algorithm that has a constant success probability; thus, we may need to run the algorithm for $O(\log n)$ times to guarantee that it successes with high probability. The derandomization of our subroutine is presented in Appendix A. Our algorithm for the rooted-connectivity augmentation is presented in Algorithm 1.

Algorithm 1 Rooted-Connectivity Augmentation.

Require: : An input graph G and an ℓ-rooted-connected graph H_ℓ
Ensure: : An $(\ell+1)$-rooted-connected graph $H_{\ell+1}$
1: Initialize $H_{\ell+1} := H_\ell$.
2: **repeat**
3: Find an optimal solution $\mathbf{x}$ to $\mathsf{LP}^{aug}(\ell)$.
4: Compute cores and their corresponding Halo-sets in $H_{\ell+1}$.
5: Find a subset of edges E' that covers at least $1/9$ fraction of the Halo-families whose cost is at most $4\sum_{e\in E_+} c_e x_e$.
6: Update $H_{\ell+1} := H_{\ell+1} + E'$.
7: **until** The graph $H_{\ell+1}$ has no deficient set (and thus has no core).
8: **return** $H_{\ell+1}$

One may observe that the covering problem in our setting is different from that in the usual Set Cover problem as after we add edges to cover γ fraction of the Halo-families, it is not guaranteed that the number of Halo-families will be decreased by a factor γ. This is because some of the deficient sets in the previous iterations may become new cores in the solution subgraph. Fortunately, we have a key property that any new core that was not contained in any Halo-families must contain at least two old cores. As a result, we can promise a factor $(1-\gamma/2)$ decrease. Please see Figure 1 for illustration. The subsets C_1 and C_2 are two cores covered by e_1 and e_2, respectively. After adding two edges, C_1 and C_2 are no longer a deficient set. Now the deficient set $C_3 \supseteq C_1 \bigcup C_2$ becomes a new core, which contains two old cores.

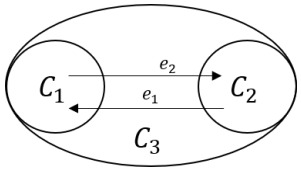


Figure 1 After adding edges e_1 and e_2 to cover C_1, C_2, a new core C_3 appear. The new core C_3 must contain at least two old cores.

▶ **Lemma 9** (The number of cores decreases by a constant factor). *Let H be the current solution subgraph whose number of cores is ν, and let E' be a set of edges that covers at least γ fraction of the Halo-families in H. Then the number of cores in $H \cup E'$ is at most $(1-\gamma/2)\nu$. In particular, the number of cores in the graph $H \cup E'$ decreases by a constant factor, provided that γ is a constant.*

Proof. Let us count the number of cores in the graph $H \cup E'$. Consider any core C in $H \cup E'$. If C is a member of some Halo-families $\mathsf{Halo}(C')$ in H, then we know that $\mathsf{Halo}(C')$ is not covered by E'. Thus, there can be at most $(1-\gamma)\nu$ cores of this type.

Next assume, otherwise, that C is not a member of any Halo-family in H. Then, by definition, C must contain at least two cores in H. Notice that, for every core C' in H that is contained in C, all of the deficients in $\mathsf{Halo}(C')$ must be covered by E'. Suppose not. Then there exists a deficient set U in $\mathsf{Halo}(C')$ that is not covered by E'. Since U interesects C on the terminal set, Lemma 2 implies that $U \cap C$ is also a deficient set. By Lemma 3, any two cores are disjoint on the terminal set, which means that $U \cap C$ is strictly contained in C (because C contains another core C'' distinct from C'). The existence of $U \cap C$ contradicts the fact that C is a core in $H \cup E'$. Thus, we conclude that $H \cup E'$ has at most $(\gamma/2)\nu$ cores of this type.

Summing it up, the total number of cores in $H \cup E'$ is at most $(1-\gamma/2)\nu$ as claimed. ◀

It follows as a corollary that our algorithm terminates within $O(\log q)$ iterations.

▶ **Corollary 10.** *The number of iterations of our algorithm is at most $O(\log q)$, where q is the number of terminals.*

By Corollary 10, our algorithm for rooted-connectivity augmentation terminates with in $O(\log q)$, and each round, we buy a set of edges whose cost is at most $4\sum_{e\in E_+} c_e x_e$; see Section 5. Therefore, the total cost incurred by our algorithm is at most $O(\log q)$ times the optimal LP solution, implying an LP-based $O(\log q)$-approximation algorithm as required by Lemma 8. The following lemma then follows immediately.

▶ **Lemma 11.** *Consider the problem of augmenting the rooted-connectivity of a directed graph from ℓ to $\ell+1$ when an input graph is quasi-bipartite. There exists a polynomial-time algorithm that gives a feasible solution whose cost at most $O(\log q)$ that of the optimal solution to the standard LP-relaxation. In particular, there exists a polynomial-time LP-based $O(\log q)$-approximation algorithm for the problem.*

▶ Remark. Lastly, we remark that one may simply cover all the Halo-families in each iteration. However, the number of rounds the randomized algorithm required will be at least $O(\log q)$, meaning that the total number of iterations is $O(\log^2 q)$. Consequently, this implies that the algorithm has to pay a factor $O(\log^2 q)$ in the approximation ratio. We avoid the extra $O(\log q)$ factor by covering only a constant fraction of the Halo-families.

4.3 Correctness and Overall Analysis

First, to prove the feasibility of the solution subgraph, it suffices to show that the rooted-connectivity of the solution subgraph increasess by at least one in each connectivity augmentation step. This simply follows by the stopping condition of the Halo-set decomposition method runs until there exists no core in the graph (and thus no deficient sets). It then follows by Menger's theorem that the number of edge-disjoint paths from the root vertex r to each terminal $t \in T$ must be increased by at least one.

Next we analyze the cost. By Lemma 11, the approximation factor incurred by Algorithm 1 is $O(\log q)$, and it also bounds the integrality gap of $\mathsf{LP}^{aug}(\ell)$. Consequently, letting OPT_k denote the cost of an optimal solution to k-DST, by Lemma 8, the total expected cost incurred by the algorithm is then

$$\sum_{\ell=1}^{k} O(\log q) \cdot \mathsf{Val}(\mathsf{LP}^{aug}(\ell)) = O(\log q) \cdot \left(\sum_{\ell=1}^{k} \frac{1}{k-\ell+1} \right) \cdot \mathsf{Val}(\mathsf{LP}(k))$$
$$= O(\log q \log k) \cdot \mathsf{OPT}_k.$$

This completes the proof of Theorem 1.

5 Covering Halo-Families via Set Cover

In this section, we present our subroutine for covering the Halo-families that arose from the Rooted-Aug problem. As mentioned in the introduction, the key ingredient in our algorithm is the reduction from the problem of covering Halo-families to the Set Cover problem. However, our instance of the Set Cover problem has an exponential number of subsets, which more resemblances to an instance of the *Facility Location* problem. To prove our result, one route would be using *Facility Location* as an intermediate problem in the presentation. However, we prefer to directly apply a reduction to the Set Cover problem to avoid confusing the readers.

5.1 The Reduction to Set Cover and Algorithm

As an overview, our reduction follows from simple observations.

- (P1) For any minimal subset of edges that covers a Halo-family $\mathsf{Halo}(C)$, there is only one edge e that has head in $\mathsf{Halo}(C)$ and tail outside. Let us say e is outer-cover $\mathsf{Halo}(C)$ since it is coming from the outside of the family.
- (P2) Any edge can be contained in at most one $\mathsf{Halo}(C)$. i.e., there is at most one halo-family $\mathsf{Halo}(C)$ such that both head and tail of e are contained in $H(C)$. (From Lemma 4.)
- (P3) An LP for covering a single Halo-family is integral.

We remark that while Properties (P1) and (P3) hold in general instances of k-DST, Property (P2) holds only in quasi-bipartite graphs.

Now an instance of the Set Cover problem can be easily deduced. We define each Halo-family $\mathsf{Halo}(C)$ as an element, and we define each edge e as a subset. However, we may have multiple subsets corresponding to the same edge e as it may serve as an "outer-cover" for many Halo-families. Thus, we need to enumerate all the possible collections of Halo-families that are outer-covered by e. We avoid getting exponential number of subsets by using the solution from an LP (for the connectivity augmentation problem) as a guideline.

Before proceeding, we need to formally define some terminologies. Let $\hat{G}$ be the current solution subgraph. We say that an edge e *outer-covers* a Halo-family $\mathsf{Halo}(C)$ if the head of e is in $H(C)$ and the tail is not in $H(C)$ and that there exists a subset of edges $E' \subseteq E_+ - E(\hat{G})$ such that (1) both endpoints of every edge in E' are contained in $H(C)$ and (2) the set of edges $E' \cup \{e\}$ covers $\mathsf{Halo}(C)$.

For each Halo-family $\mathsf{Halo}(C)$, we define the set of edges I_C^e to be the minimum-cost subset of edges $E' \subseteq E_+ - E(\hat{G})$ whose both endpoints are in $H(C)$ and that $E' \cup \{e\}$ covers $\mathsf{Halo}(C)$, and we denote the cost of I_C^e by σ_C^e. We may think that σ_C^e is the cost for covering

$\mathsf{Halo}(C)$ given that e has been taken for free. We use the notation $E[C]$ to mean the set of edges whose both endpoints are contained in the Halo-set $H(C)$. We denote the cost of the fractional solution restricted to $E[C]$ by $\mathsf{cost}_x(E(C)) = \sum_{e \in E[C]} c_e x_e$.

Our reduction is as follows. Let H be the current solution subgraph. For each core C in H, we define an element C. For each edge $e \in E_+ - E(H)$, we define a subset S_e by adding to S_e an element C if $\sigma_C^e \leq \mathsf{cost}_x(E[C])$. This completes a reduction. Notice that the resulting instance of the Set Cover problem has polynomial size. To show that our reduction runs in polynomial-time, we need to give a polynomial-time algorithm for computing σ_C^e, which we defer to Section 5.4. Here we leave a forward reference to Lemma 15. Our algorithm covers a constant fraction of the collection of Halo-families by simply picking each edge e with probability x_e and add all the edges I_C^e, for all cores $C \in S_e$, to the solution subgraph; if a core C is outer-covered by two picked edges, then we add only one edge-set I_C^e. We claim that the set of edges chosen by our algorithm covers at least $1/9$ fraction of the Halo-families, while paying a cost of at most four times the optimum (with a constant probability). In particular, we prove the following lemma.

▶ **Lemma 12.** *With constant probability, the above algorithm covers at least* $1/9$ *fraction of the collection of Halo-families, and the cost of the of the edges chosen by the algorithm has cost at most* $4\sum_{e \in E_+} c_e x_e$*. In particular, the algorithm partially covers the collection of the Halo-families, while paying the cost of at most constant times the optimum.*

To prove Lemma 12, we need to show that the fractional solution defined by $\{x_e\}_{e \in E+}$ is (almost) feasible to the Set Cover instance, which then implies that the set of edges we bought covers a constant fraction of the Halo-families with probability at least $2/3$. Then we will show that the cost of the fractional solution to the Set Cover instance is at most twice that of the optimal solution to $\mathsf{LP}^{aug}(\ell)$, thus implying that we pay at most six times the optimum with probability $2/3$.

To be more precise, we show in Section 5.2 that our algorithm covers at least $1/3$ fraction of the Halo-families in expectation, meaning that we cover less than $1/9$ fraction with probability at most $1/3$. Then we show in Section 5.3 that the expected cost incurred by our algorithm is $2\sum_{e \in E_+} c_e x_e$, thus implying that we pay more than six times that of the LP with probability at most $1/3$. Applying the union bound, we conclude that our algorithm covers at least $1/9$ fraction of the Halo-families, while paying the cost of at most six times the optimal LP solution with probability at least $1/3$. (Note that in Section 5.3, we show a slightly stronger statement that the cost incurred by our algorithm is $4\sum_{e \in E_+} c_e x_e$ with probability at least $2/3$.) To finish our proof, we proceed to prove the above two claims and then prove the structural properties used in the forward references.

5.2 Partial Covering

We show in this section that our algorithm covers at least $1/9$ fraction of the Halo-families with probability at least $1/3$

First, we show that the LP variable defined by x_e is almost feasible to the LP-relaxation of the Set Cover problem. We note that our proof will need a forward reference to Lemma 14.

▶ **Lemma 13.** *The LP variable* $\{y_e\}_{e \in E_+}$*, where* $y_e = \min\{1, 2x_e\}$ *for all edges* $e \in E_+$ *is feasible to the Set Cover instance. That is, for any core* C *in the graph,*

$$\sum_{e \in E_+ : C \in S_e} x_e \geq 1/2.$$

Proof. Consider a core C, which corresponds to an element in the Set Cover instance. We take the set of edges incident to its Halo-set $H(C)$, and find a minimal vectors $\{x'_e\}_{e\in E_+}$ such that $\{x'_e\}_{e\in E_+}$ fractionally covers the Halo-family $\mathsf{Halo}(C)$ and $x'_e \le x_e$ for all edges $e \in E_+$. (Note that by minimality we mean that, for any edge e and any $\epsilon > 0$, decreasing the value of x'_e by ϵ results in an infeasible solution.) By Lemma 14, we have $\sum_{e\in\delta^{in}(H(C))} x'_e = 1$, i.e., the total weight of the LP value of edges incoming to $H(C)$ is exactly one.

Next consider the following LP.

$$\mathsf{LP}^{halo} = \begin{cases} \min & \sum_{e'\in E_+(H(C))} c_{e'}x_{e'} & \\ \text{s.t} & \sum_{e'\in\delta^{in}_{E_+}(U)} x_{e'} \ge 1 & \forall U \in \mathsf{Halo}(C) \\ & 0 \le x_{e'} \le 1 & \forall e \in E_+(H(C)) \end{cases}$$

By Lemma 2, we know that both the intersection and union of any two deficient sets in $\mathsf{Halo}(C)$ are also deficient sets in $\mathsf{Halo}(C)$. This means that the Halo-family $\mathsf{Halo}(C)$ is an intersecting family. It then follows from the result of Frank [21] that the above LP is *Totally Dual Integral*, which means that any convex point of its polytope is an integral solution (including the optimal one). Since $\{x'_e\}_{e\in E'}$ is a feasible solution to LP^{halo}, it can be written as a convex combination of integral vectors in the polytope, i.e.,

$$\mathbf{x} = \sum_{i=1}^{w} \lambda_i \mathbf{z}^i, \text{ where } \sum_{i=1}^{w} \lambda_i = 1.$$

Let F_i be the set of edges induced by each integral vector $\mathbf{z}^i$ (i.e., F_i is the support of $\mathbf{z}^i$). Since the LP requires $H(C)$ to have at least one incoming edge, we deduce that, for each F_i, there exists one edge $e_i \in F_i$ entering $H(C)$.

Now we compare the cost of $\sigma_C^{e_i}$ to the cost of $F_i - \{e_i\}$. By minimality of $\sigma_C^{e_i}$, we know that $\sigma_C^{e_i} \le \mathsf{cost}(F_i - \{e_i\})$ for all $i = 1, \ldots, w$. We recall that we add a core C to the set S_{e_i} only if $\sigma_C^{e_i} \le \mathsf{cost}_x(E[C])$. Since $\mathsf{cost}_{x'}(E[C])$ is the convex combination of $\mathbf{Z}^i$, at least half of the F_i (w.r.t. to the weight λ_i) must have $\sigma_C^{e_i} \le \mathsf{cost}(F_i - \{e_i\}) \le \mathsf{cost}_x(E[C])$; that is, $\sum_{i:\sigma_C^e \le \mathsf{cost}(F_i-\{e_i\})} \lambda_i \ge 1/2$. Therefore, we conclude that the sum of y_{e_i} over all e_i such that $\sigma_C^{e_i} \le \mathsf{cost}_x(E[C])$ is at least one, thus proving the lemma. ◀

We remark that we may define the Set Cover instance so that $\{x_e\}_{e\in E_+}$ is exactly a feasible solution to the LP for the Set Cover problem by using the integer decomposition as in the proof of Lemma 13. However, we choose to present it this way to keep the reduction simple.

Now we finish the proof of our claim. Consider a core C. Note that by construction, every time we pick an edge e, we also add the set of edges I_C^e, for each $C \in S_e$ (recall that $I_C^e \cup \{e\}$ covers $\mathsf{Halo}(C)$). Thus, the probability that the algorithm picks no edges e such that $C \in S_e$ is

$$\Pi_{e\in E_+:C\in S_e}(1 - x_e) \le \exp\left(-\sum_{e\in E_+} x_e\right) \le \exp(-1/2) \le \frac{2}{3}.$$

The first inequality follows because $1 - x \le \exp(-x)$, for $0 < x \le 1$. That is, the probability that the algorithm does not cover a core C is at most $2/3$, which means that the expected fraction of Halo-families covered by our algorithm is at least $1/3$. Applying Markov's inequality, we conclude that with probability at least $2/3$ our algorithms covers at least $1/9$ fraction of the Halo-families. Our algorithm can be derandomized using the method of conditional expectation. Please see Appendix A for details.

5.3 Cost Analysis

Now we analyze the expected cost of the edges we add to the solution subgraph. We classify the cost incurred by our algorithm into two categories. The first case is the set of edges e that we pick with probability x_e. The expected cost of this case is $\sum_{e \in E_+} c_e x_e$. Applying Markov's inequality, we have that with probability at least $2/3$ the cost incurred by the edges of this case is at most $3 \sum_{e \in E_+} c_e x_e$.

The second case is the set of edges corresponding to each subset S_e whose edge e is added to the solution. By construction, a core C is added to S_e only if $\mathsf{cost}_x(E[C])$ is greater than σ^e_C (i.e., the cost of the set of edges I^e_C). We recall that we also add one set of edges I^e_C to the solution if there are more than one edges e such that $C \in S_e$ are chosen. As the set of edges $E[C]$ and $E[C']$ are disjoint for any two cores $C \neq C'$ (please see the forward reference to Lemma 4), we conclude that the cost incurred by the edges of this case is at most $\sum_{e \in E_+} c_e x_e$ (regardless of the choices of the edges randomly picked in the previous step). Therefore, with probability at least $2/3$ the cost of edges chosen by our algorithm is at most $4 \sum_{e \in E_+} c_e x_e$.

5.4 Structural Properties of the LP solution

We devote this last subsection to prove properties (P1) to (P3) and all the forward references as discussed earlier. Property (P3) simply follows from the fact that the intersection and union of any two members of a Halo-family $\mathsf{Halo}(C)$ are also members of $\mathsf{Halo}(C)$, which means that the polytope of the problem of covering $\mathsf{Halo}(C)$ is integral due to the result of Frank [21]. Thus, we are left to prove the property (P1) and (P2) and to present a polynomial-time algorithm for computing σ^e_C, which thus complete the proof that our reduction can be done in polynomial time.

First, we prove Property (P1), which allows us to reduce the instance of the problem of covering Halo-families to a Set Cover instance.

▶ **Lemma 14** (Unique Entering Edge in Minimal Cover)**.** *Consider a minimal fractional cover x of a Halo-family $\mathsf{Halo}(C)$. That is, x is a feasible solution to LP^{halo} whose collection of deficient sets is defined by $\mathsf{Halo}(C)$, and decreasing the value x_e of any edge $e \in E_+$ results in an infeasible solution. It holds that $\sum_{e \in \delta^{in}_{E_+}(H(C))} x_e = 1$. Thus, for an integral solution E', there is exactly one edge $e \in E'$ entering the Halo-set $H(C)$.*

Proof. Assume for a contradiction that $\sum_{e \in \delta^{in}_{E_+}(H(C))} x_e > 1$. By the minimality of x, for any edge $e \in \delta^{in}_{E_+}(H(C))$, there exists a deficient set $W_e \in \mathsf{Halo}(C)$ such that $\sum_{e \in \delta^{in}_{E_+}(W_e)} x_e = 1$. We choose W_e to be the maximum inclusionwise such set and call it the witness set of e.

Now we take two distinct witness sets W_e and $W_{e'}$, for $e \neq e'$. By Lemma 5, both $W_e \cap W_{e'}$ and $W_e \cup W_{e'}$ are deficient sets in $\mathsf{Halo}(C)$. Let us abuse the notation of x. For any subset of vertices $S \subseteq V(G)$, let $x(S) = \sum_{e \in \delta^{in}_{E_+}(S)} x_e$. The function $x(S)$ is known to be submodular [24], meaning that

$$2 = x(W_e) + x(W_{e'}) \geq x(W_e \cap W_{e'}) + x(W_e \cup W_{e'}) \geq 2.$$

The last inequality follows because $\{x\}_{e \in E_+}$ fractionally covers $\mathsf{Halo}(C)$, which then implies that $x(W_e \cap W_{e'}) = x(W_e \cup W_{e'}) = 1$. But, this contradicts the choice of W_e (and also $W_{e'}$) because $W_e \cup W_{e'}$ is a deficient set in $\mathsf{Halo}(C)$ strictly containing W_e in which the conditions $x(W_e \cup W_{e'}) = 1$ and $e \in \delta^{in}_{E_+}(W_e \cup W_{e'})$ hold. ◀

Next we prove Property (P2), which allows us to upper bound the cost incurred by the main algorithm.

Finally, we show that σ^e_C can be computed in polynomial time.

▶ **Lemma 15.** *For any core $C \in \mathcal{C}$ and an edge $e \in E(G)$, the set of edges I^e_C and, thus, its cost σ^e_C can be computed in polynomial time. Moreover, the value of σ^e_C is equal to the optimal value of the corresponding covering LP given below.*

$$\mathsf{LP}^{cover} = \begin{cases} \min & \sum_{e' \in E_+(H(C))} c_{e'} x_{e'} & \\ s.t & \sum_{e' \in \delta^{in}_{E_+}(U)} x_{e'} \geq 1 & \forall U \in \mathsf{Halo}(C) \\ & 0 \leq x_{e'} \leq 1 & \forall e \in E_+(H(C)) \\ & x_e = 1 & \end{cases}$$

Proof. Consider the Halo-family $\mathsf{Halo}(C)$. By Lemma 5, the union and intersection of any deficient sets $U, W \in \mathsf{Halo}(C)$ are also deficient sets in $\mathsf{Halo}(C)$. This means that $\mathsf{Halo}(C)$ is an *intersecting family*. It is known that the standard LP for covering an intersecting family is integral (see, e.g., [21]), which implies that we can compute σ^e_C and its corresponding set of edges I^e_C in polynomial time by solving LP^{cover}.

Alternatively, we may compute σ^e_C combinatorially using an efficient minimum-cost $(\ell+1)$-flow algorithm. In particular, we construct an s^*, t^*-flow network by setting the costs of edges in $\delta^{in}_{H_{\ell+1}}(H(C)) \cup \{e\}$ to zero, adding a source s^* connecting to $\ell+1$ edges entering $\mathsf{Halo}(C)$ (which consists of ℓ edges from $\delta^{in}_{H_{\ell+1}}(H(C))$ plus the edge e) and then picking an arbitrary terminal $t^* \in C$ as a sink. All the edges not in $E(H(C))$ except $\delta^{in}_{H_{\ell+1}}(H(C)) \cup \{e\}$ are removed. Applying Manger's theorem, it can be seen that every $(\ell+1)$-flow in this s^*, t^*-flow network corresponds to a feasible solution to the covering problem with the same cost. This gives a polynomial-time algorithm for computing σ^e_C and I^e_C as desired. ◀

6 Conclusion and Open Problems

We have presented our $O(\log q \log k)$-approximation algorithm for k-DST when an input graph is quasi-bipartite. This is the first polylogarithmic approximation algorithm for k-DST for arbitrary k that does not require an additional assumption on the structure of the optimal solution. In addition, our result implies that k-DST in quasi-bipartite graphs is equivalent to the Set Cover problem when $k = O(1)$.

Lastly, we conclude our paper with some open problems. A straightforward question is whether there exists a non-trivial approximation algorithm for k-DST for $k \geq 3$ in general case or for a larger class of graphs (perhaps, in quasi-polynomial-time). Another interesting question is whether our randomized rounding technique, which consists of dependent rounds of a randomized rounding algorithm for the Set Cover problem, can be applied without connectivity augmentation. If this is possible, it will give $O(\log k)$ improvements upon the approximation ratios for approximating many problems whose the best known algorithms are based on the Halo-Set decomposition technique.

References

1 Noga Alon and Joel H. Spencer. *The Probabilistic Method.* Wiley Publishing, 4th edition, 2016.

2 Mihir Bellare, Shafi Goldwasser, Carsten Lund, and A. Russeli. Efficient probabilistically checkable proofs and applications to approximations. In S. Rao Kosaraju, David S. Johnson, and Alok Aggarwal, editors, *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 294–304. ACM, 1993. doi:10.1145/167088.167174.

3 Kristóf Bérczi and Erika Renáta Kovács. A note on strongly edge-disjoint arborescences. In *7th Japanese-Hungarian Symposium on Discrete Mathematics and its Applications*, pages 10–18, 2011. Also, in EGRES Technical Reports series TR-2011-04.

4 Dimitris Bertsimas and Rakesh Vohra. Rounding algorithms for covering problems. *Mathematical Programming*, 80:63–89, 1998. doi:10.1007/BF01582131.

5 Jaroslaw Byrka, Fabrizio Grandoni, Thomas Rothvoß, and Laura Sanità. Steiner tree approximation via iterative randomized rounding. *J. ACM*, 60(1):6:1–6:33, 2013. Preliminary version in STOC'10. doi:10.1145/2432622.2432628.

6 Deeparnab Chakrabarty, Nikhil R. Devanur, and Vijay V. Vazirani. New geometry-inspired relaxations and algorithms for the metric steiner tree problem. *Math. Program.*, 130(1):1–32, 2011. Preliminary version in IPCO'08. doi:10.1007/s10107-009-0299-0.

7 Tanmoy Chakraborty, Julia Chuzhoy, and Sanjeev Khanna. Network design for vertex connectivity. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 167–176. ACM, 2008. doi:10.1145/1374376.1374403.

8 Parinya Chalermsook, Fabrizio Grandoni, and Bundit Laekhanukit. On survivable set connectivity. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 25–36. SIAM, 2015. doi:10.1137/1.9781611973730.3.

9 Moses Charikar, Chandra Chekuri, To-Yat Cheung, Zuo Dai, Ashish Goel, Sudipto Guha, and Ming Li. Approximation algorithms for directed steiner problems. *J. Algorithms*, 33(1):73–91, 1999. doi:10.1006/jagm.1999.1042.

10 Chandra Chekuri and Nitish Korula. A graph reduction step preserving element-connectivity and applications. In Susanne Albers, Alberto Marchetti-Spaccamela, Yossi Matias, Sotiris E. Nikoletseas, and Wolfgang Thomas, editors, *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part I*, volume 5555 of *Lecture Notes in Computer Science*, pages 254–265. Springer, 2009. doi:10.1007/978-3-642-02927-1_22.

11 Joseph Cheriyan and Bundit Laekhanukit. Approximation algorithms for minimum-cost k-(S,T) connected digraphs. *SIAM J. Discrete Math.*, 27(3):1450–1481, 2013. doi:10.1137/100818728.

12 Joseph Cheriyan, Bundit Laekhanukit, Guyslain Naves, and Adrian Vetta. Approximating rooted steiner networks. *ACM Trans. Algorithms*, 11(2):8:1–8:22, 2014. Preliminary version in SODA'12. doi:10.1145/2650183.

13 Julia Chuzhoy and Sanjeev Khanna. Algorithms for single-source vertex connectivity. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 105–114. IEEE Computer Society, 2008. doi:10.1109/FOCS.2008.63.

14 Julia Chuzhoy and Sanjeev Khanna. An $o(k^3 \log n)$-approximation algorithm for vertex-connectivity survivable network design. *Theory of Computing*, 8(1):401–413, 2012. Preliminary version in FOCS'09. doi:10.4086/toc.2012.v008a018.

15 Irit Dinur and David Steurer. Analytical approach to parallel repetition. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633. ACM, 2014. doi:10.1145/2591796.2591884.

16 Yevgeniy Dodis and Sanjeev Khanna. Design networks with bounded pairwise distance. In Jeffrey Scott Vitter, Lawrence L. Larmore, and Frank Thomson Leighton, editors, *Proceedings of the Thirty-First Annual ACM Symposium on Theory of Computing, May 1-4, 1999, Atlanta, Georgia, USA*, pages 750–759. ACM, 1999. doi:10.1145/301250.301447.

17 Jittat Fakcharoenphol and Bundit Laekhanukit. An $o(\log^2 k)$-approximation algorithm for the k-vertex connected spanning subgraph problem. *SIAM J. Comput.*, 41(5):1095–1109, 2012. Preliminary version in STOC'08. doi:10.1137/110855910.

18 Uriel Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. Preliminary version in STOC'96. doi:10.1145/285055.285059.

19 Moran Feldman, Guy Kortsarz, and Zeev Nutov. Improved approximating algorithms for directed steiner forest. 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 922–931. SIAM, 2009. doi:10.1137/1.9781611973068.

20 Lisa Fleischer, Kamal Jain, and David P. Williamson. Iterative rounding 2-approximation algorithms for minimum-cost vertex connectivity problems. *J. Comput. Syst. Sci.*, 72(5):838–867, 2006. doi:10.1016/j.jcss.2005.05.006.

21 András Frank. Kernel systems of directed graphs. *Acta Sci. Math.(Szeged)*, 41(1-2):63–76, 1979.

22 András Frank. Increasing the rooted-connectivity of a digraph by one. *Math. Program.*, 84(3):565–576, 1999. doi:10.1007/s101070050040.

23 András Frank and Tibor Jordán. Directed vertex-connectivity augmentation. *Math. Program.*, 84(3):537–553, 1999. doi:10.1007/s101070050038.

24 András Frank and Tibor Jordán. Graph connectivity augmentation. In *Handbook of Graph Theory, Combinatorial Optimization, and Algorithms*, pages 315–348. CRC Press, 2016.

25 Zachary Friggstad, Jochen Könemann, Young Kun-Ko, Anand Louis, Mohammad Shadravan, and Madhur Tulsiani. Linear programming hierarchies suffice for directed steiner tree. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, pages 285–296, 2014. doi:10.1007/978-3-319-07557-0_24.

26 Zachary Friggstad, Jochen Könemann, and Mohammad Shadravan. A logarithmic integrality gap bound for directed steiner tree in quasi-bipartite graphs. In Rasmus Pagh, editor, *15th Scandinavian Symposium and Workshops on Algorithm Theory, SWAT 2016, June 22-24, 2016, Reykjavik, Iceland*, volume 53 of *LIPIcs*, pages 3:1–3:11. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.SWAT.2016.3.

27 Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. *J. Algorithms*, 37(1):66–84, 2000. Preliminary version in SODA'98. doi:10.1006/jagm.2000.1096.

28 Loukas Georgiadis and Robert E. Tarjan. Dominator tree certification and divergent spanning trees. *ACM Transactions on Algorithms*, 12(1):11, 2016. Preliminary version in SODA'05. doi:10.1145/2764913.

29 Rohan Ghuge and Viswanath Nagarajan. A quasi-polynomial algorithm for submodular tree orienteering in directed graphs. *CoRR*, abs/1812.01768, 2018. arXiv:1812.01768.

30 Michel X. Goemans, Andrew V. Goldberg, Serge A. Plotkin, David B. Shmoys, Éva Tardos, and David P. Williamson. Improved approximation algorithms for network design problems. In Daniel Dominic Sleator, editor, *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms. 23-25 January 1994, Arlington, Virginia, USA.*, pages 223–232. ACM/SIAM, 1994. URL: http://dl.acm.org/citation.cfm?id=314464.314497.

31 Michel X. Goemans, Neil Olver, Thomas Rothvoß, and Rico Zenklusen. Matroids and integrality gaps for hypergraphic steiner tree relaxations. 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 1161–1176. ACM, 2012. doi:10.1145/2213977.2214081.

32 Fabrizio Grandoni and Bundit Laekhanukit. Surviving in directed graphs: a quasi-polynomial-time polylogarithmic approximation for two-connected directed steiner tree. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 420–428. ACM, 2017. doi:10.1145/3055399.3055445.

33 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. In *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, Phoenix, AZ, USA, June 23-26, 2019.*, pages 253–264, 2019. doi:10.1145/3313276.3316349.

34 Tomoya Hibi and Toshihiro Fujito. Multi-rooted greedy approximation of directed steiner trees with applications. *Algorithmica*, 74(2):778–786, 2016. doi:10.1007/s00453-015-9973-1.

35 Kamal Jain. A factor 2 approximation algorithm for the generalized steiner network problem. *Combinatorica*, 21(1):39–60, 2001. Preliminary version in FOCS'01. doi:10.1007/s004930170004.

36 Philip N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995. Preliminary version in IPCO'93. doi:10.1006/jagm.1995.1029.

37 Jochen Könemann, David Pritchard, and Kunlun Tan. A partition-based relaxation for steiner trees. *Math. Program.*, 127(2):345–370, 2011. doi:10.1007/s10107-009-0289-2.

38 Guy Kortsarz, Robert Krauthgamer, and James R. Lee. Hardness of approximation for vertex-connectivity network design problems. *SIAM J. Comput.*, 33(3):704–720, 2004. Preliminary version in APPROX'02. doi:10.1137/S0097539702416736.

39 Guy Kortsarz and Zeev Nutov. Approximating k-node connected subgraphs via critical graphs. *SIAM J. Comput.*, 35(1):247–257, 2005. Preliminary version in STOC'04. doi:10.1137/S0097539703435753.

40 Guy Kortsarz and David Peleg. Approximating shallow-light trees (extended abstract). In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, 5-7 January 1997, New Orleans, Louisiana, USA.*, pages 103–110, 1997. URL: http://dl.acm.org/citation.cfm?id=314161.314191.

41 Bundit Laekhanukit. Parameters of two-prover-one-round game and the hardness of connectivity problems. 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 1626–1643. SIAM, 2014. doi:10.1137/1.9781611973402.118.

42 Bundit Laekhanukit. An improved approximation algorithm for the minimum cost subset k-connected subgraph problem. *Algorithmica*, 72(3):714–733, 2015. Preliminary version in ICALP'11. doi:10.1007/s00453-014-9869-5.

43 Bundit Laekhanukit. Approximating directed steiner problems via tree embedding. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 74:1–74:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016. doi:10.4230/LIPIcs.ICALP.2016.74.

44 Dana Moshkovitz. The projection games conjecture and the np-hardness of ln n-approximating set-cover. *Theory of Computing*, 11:221–235, 2015. Preliminary version in APPROX'12. doi:10.4086/toc.2015.v011a007.

45 Zeev Nutov. Approximating minimum power covers of intersecting families and directed edge-connectivity problems. *Theor. Comput. Sci.*, 411(26-28):2502–2512, 2010. Preliminary version in APPROX'06. doi:10.1016/j.tcs.2010.03.009.

46 Zeev Nutov. Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Trans. Algorithms*, 9(1):1:1–1:16, 2012. Preliminary version in FOCS'09. doi:10.1145/2390176.2390177.

47 Zeev Nutov. Approximating subset k-connectivity problems. *J. Discrete Algorithms*, 17:51–59, 2012. Preliminary version in WAOA'11. doi:10.1016/j.jda.2012.08.001.

48 Zeev Nutov. Approximating minimum-cost edge-covers of crossing biset-families. *Combinatorica*, 34(1):95–114, 2014. Preliminary version in SODA'09. doi:10.1007/s00493-014-2773-4.

49 Zeev Nutov. Erratum: Approximating minimum-cost connectivity problems via uncrossable bifamilies. *ACM Trans. Algorithms*, 14(3):37:1–37:8, 2018. doi:10.1145/3186991.

50 Harald Räcke. Optimal hierarchical decompositions for congestion minimization in networks. In Cynthia Dwork, editor, *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 255–264. ACM, 2008. doi:10.1145/1374376.1374415.

51 Sridhar Rajagopalan and Vijay V. Vazirani. On the bidirected cut relaxation for the metric steiner tree problem. 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 742–751. ACM/SIAM, 1999. URL: http://dl.acm.org/citation.cfm?id=314500.314909.

52 Romeo Rizzi. On rajagopalan and vazirani's 3/2-approximation bound for the iterated 1-steiner heuristic. *Inf. Process. Lett.*, 86(6):335–338, 2003. doi:10.1016/S0020-0190(03)00210-2.

53 Gabriel Robins and Alexander Zelikovsky. Improved steiner tree approximation in graphs. In David B. Shmoys, editor, *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 770–779. ACM/SIAM, 2000. URL: http://dl.acm.org/citation.cfm?id=338219.338638.

54 Thomas Rothvoß. Directed steiner tree and the lasserre hierarchy. *CoRR*, abs/1111.5473, 2011. arXiv:1111.5473.

55 David P. Williamson, Michel X. Goemans, Milena Mihail, and Vijay V. Vazirani. A primal-dual approximation algorithm for generalized steiner network problems. *Combinatorica*, 15(3):435–454, 1995. Preliminary version in STOC'93. doi:10.1007/BF01299747.

56 Alexander Zelikovsky. A series of approximation algorithms for the acyclic directed steiner tree problem. *Algorithmica*, 18(1):99–110, 1997. doi:10.1007/BF02523690.

A Derandomization

In this section, we present a derandomization of our algorithm in Section 5 using the method of conditional expectation [1]. We will mostly follow the proof presented in the work of Bertsimas and Vohra [4] who gave a derandomized technique for the randomized scheme for the Set Cover problem.

In more detail, first observe that the cost incurred by our algorithm comes from two parts. The firt part is the cost of edges e that we pick with probabilty x_e, and the second part is the cost of edges I_e^C in which the edge e is chosen. For the second part, our algorithm guarantees that, for each core C, only one set of edges I_e^C will be added to the solution. Thus, by the construction of S_e and Lemma 4, the cost incurred by this part is $\sum_{e\in E_+} c_e x_e$ regardless of the choices of the edges e added to the solution from the first part.

Hence, it suffices to show that there exists a deterministic algorithm that pick a set of edges E' that outter-covers at least $1/3$ fraction of the Halo-families, while paying the cost at most $\sum_{e\in E_+} c_e x_e$.

Let $\mathcal{C}$ be the collection of all the cores in the current solution subgraph. For a given set of edges $E' \subseteq E_+$, we define a function $\tau_C \in \{0,1\}$ for each Halo-family $\mathsf{Halo}(C)$ to indicate whether $\mathsf{Halo}(C)$ is covered by some edge in E', and we define a function $\mathbb{I}(\vec{\tau})$ to indicate whether E' outer-covers at least $1/9$ fraction of the Halo-families. The formal definition of these two functions are given below.

$$\tau_C(E') = \begin{cases} 1 & \text{if } E' \text{ outer-covers } \mathsf{Halo}(C) \\ 0 & \text{Otherwise} \end{cases}$$

$$\mathbb{I}(E') = \begin{cases} 1 & \sum_{C\in\mathcal{C}} \tau_C(E') < \frac{|\mathcal{C}|}{9} \\ 0 & \text{Otherwise} \end{cases}$$

Next we define the potential function:

$$\Phi(E') = \sum_{e\in E'} c_e + M \cdot \mathbb{I}(E'), \text{ where } M = 3 \sum_{e\in E_+} c_e x_e.$$

Observe that $\Phi(E') \leq M$ if E' outer-covers at least $1/9$ fraction of the Halo-families, while having the cost at most three times that of the LP solution; otherwise, $\Phi(X) > M$. Notice that, by Lemma 12, if we add each edge $e \in E_+$ to E' with probability x_e, then $\mathbb{E}[\Phi(E')] \leq M$. Thus, there exists an event that $\Phi(X) \leq M$, which will give us the desired integer solution. We then follow the method of conditional expectation (see, e.g., [1]). That is, we order edges in E_+ in an arbitrary order, say $e_1, e_2, \ldots, e_{|E_+|}$. Let E'' be the set of edges that we try to simulate the set of randomly chosen edges E'. Initially, $E_{det} = \emptyset$. Then we decide to add each edge e_i, for $i = 1, 2, \ldots, |E_+|$ to E' if $\mathbb{E}[\Phi(E')|E_{det} \cup \{e_i\} \subseteq E'] \leq \mathbb{E}[\Phi(E')|E_{det} \subseteq E']$. This way the resulting set of edges E_{det} outer-covers at least $1/9$ fraction of the Halo-families, while having the cost of at most $3\sum_{e\in E_+} c_e x_e$. Therefore, after adding the set of edges I_e^C for each core outer-covered by some edge $e \in E_{det}$, we have a set of edges that covers at least $1/9$ fraction of the Halo-families with cost at most $4\sum_{e\in E_+} c_e x_e$, i.e., with the same guarantee as desired in Lemma 12.

B Bad Example for Grandoni-Laekhanukit Tree-Embedding Approach

In [32], Grandoni and Laekhanukit proposed an approximation scheme for k-DST based on the decomposition of an optimal solution into k divergent arborescences [28, 3]. Their approach results in the first non-trivial approximation algorithm for 2-DST, and the algorithm achieves polylogarithmic approximation ratio in quasi-polynomial-time. Nevertheless, this technique meets a barrier as soon as $k \geq 3$ as it was shown in [3] that the decomposition of an optimal solution into k divergent arborescences does not exist for general graphs when $k \geq 3$. One would hope that the decomposition is still possible for some classes of graphs, e.g., quasi-bipartite graphs. We show that, unfortunately, even for the class of quasi-bipartite graphs the divergent arborescences decomposition does not exist for $k \geq 3$. The counter example of a 3-rooted-connected graph that has no 3 divergent arborescences is shown in Figure 2.

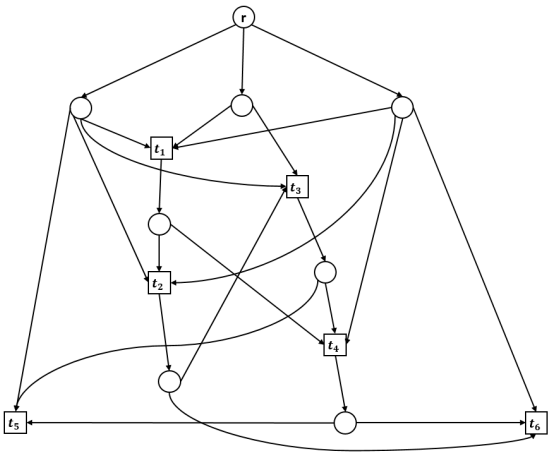


Figure 2 This figure shows an example 3-rooted-connected quasi-bipartite graph that cannot be decomposed into 3 divergent arborescences.

Weighted Maximum Independent Set of Geometric Objects in Turnstile Streams

Ainesh Bakshi
Carnegie Mellon University, Pittsburgh, PA, USA
abakshi@cs.cmu.edu

Nadiia Chepurko
MIT, Cambridge, MA, USA
nadiia@mit.edu

David P. Woodruff
Carnegie Mellon University, Pittsburgh, PA, USA
dwoodruf@cs.cmu.edu

Abstract

We study the Maximum Independent Set problem for geometric objects given in the data stream model. A set of geometric objects is said to be independent if the objects are pairwise disjoint. We consider geometric objects in one and two dimensions, i.e., intervals and disks. Let α be the cardinality of the largest independent set. Our goal is to estimate α in a small amount of space, given that the input is received as a one-pass stream. We also consider a generalization of this problem by assigning weights to each object and estimating β, the largest value of a weighted independent set. We initialize the study of this problem in the turnstile streaming model (insertions and deletions) and provide the first algorithms for estimating α and β.

For unit-length intervals, we obtain a $(2+\epsilon)$-approximation to α and β in $\text{poly}(\frac{\log(n)}{\epsilon})$ space. We also show a matching lower bound. Combined with the 3/2-approximation for insertion-only streams by Cabello and Perez-Lanterno [11], our result implies a separation between the insertion-only and turnstile model. For unit-radius disks, we obtain a $\left(\frac{8\sqrt{3}}{\pi}\right)$-approximation to α and β in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space, which is closely related to the hexagonal circle packing constant.

Finally, we provide algorithms for estimating α for arbitrary-length intervals under a bounded intersection assumption and study the parameterized space complexity of estimating α and β, where the parameter is the ratio of maximum to minimum interval length.

2012 ACM Subject Classification Theory of computation → Streaming, sublinear and near linear time algorithms; Theory of computation → Streaming models

Keywords and phrases Weighted Maximum Independent Set, Geometric Graphs, Turnstile Streams

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.64

Category APPROX

Related Version https://arxiv.org/abs/1902.10328

Funding Ainesh Bakshi and David Woodruff acknowledge support in part from NSF No. CCF-1815840.

Acknowledgements Part of this work was done while Ainesh Bakshi and David Woodruff were visiting the Simons Institute for the Theorem of Computing.

1 Introduction

Maximum Independent Set (MIS) is a fundamental combinatorial problem and in general, is NP-Hard to approximate within a $n^{1-\epsilon}$ factor, for any constant $\epsilon > 0$ [34]. We focus on the MIS problem for geometric objects: we are given as input n intervals on the real line or disks in the plane and our goal is to output the largest set of non-overlapping intervals


© Ainesh Bakshi, Nadiia Chepurko, and David P. Woodruff;
licensed under Creative Commons License CC-BY



or disks. Computing the Maximum Independent Set of intervals and disks has numerous applications in scheduling, resource allocation, cellular networks, map labellings, clustering, wireless ad-hoc networks and coding theory, where it has been extensively studied [31] [4], [50], [12], [6], [5], [1], [33], [45].

In the one dimensional setting, the MIS problem, also known as the Interval Scheduling[1] problem, has a simple greedy algorithm that picks intervals in increasing order of their right endpoint to obtain an optimal solution. The variant with weighted intervals can also be solved in polynomial time using dynamic programming, which is shown in a number of modern algorithms textbooks [21], [41]. These algorithms have considerable applications in resource allocation and scheduling, where offline and online variants have been extensively studied and we refer the reader to [42] for a survey.

In the two dimensional setting, MIS of geometric objects, such as line segments [35], rectangles [30], [37] and disks [19], is *NP-Hard*. However, in the offline setting (polynomial space), a PTAS is known for fat objects (squares, disks) and pseudo-disks [15] (who also provide a recent survey). The MIS problem for arbitrary rectangles has also received considerable attention: [13] show a $\log\log(n)$ approximation in polynomial time and [18] obtain a $(1+\epsilon)$-approximation in $n^{\text{poly}(\log(n))\epsilon^{-1}}$ time for axis-aligned rectangles.

Streaming Model. The increase in modern computational power has led to massive amounts of available data. Therefore, it is unrealistic to assume that our data fits in RAM. Instead, working with the assumption that data can be efficiently accessed in a sequential manner has led to streaming algorithms for a number of problems. Several classical problems such as heavy-hitters and l_p sampling [38], l_p estimation [39], entropy estimation [44], [20], maximum matching [43] etc. have been studied in the turnstile model and recent work has led to interesting connections with linear sketches [2].

In this paper, we study the streaming complexity of the geometric MIS problem, where the input is a sequence of n updates, either inserting a new object or deleting a previously inserted object. We assume that the algorithm has poly-logarithmic bounded memory and at the end of the stream, the algorithm should output an estimate of the (weighted) cardinality of the MIS. Since most real world scheduling applications are dynamic, and scheduling constraints expire, it is crucial to allow for both insertions and deletions, while operating in the low-space setting. Consider the following concrete application: automatic point-label conflict resolution on interactive maps [48]. In this problem, the goal is to label features (geometric objects such as points, lines and polygons) on a map such that no two features with the same label overlap. Labelling maps in visual analytic software requires such labelling to be fast and dynamic, since features can be added and removed.

1.1 Our Contributions

We provide the first algorithmic and hardness results for the Weighted Maximum Independent Set (WMIS) problem for geometric objects in *turnstile streams* (where previously inserted objects may also be deleted). The aim of our work is to understand the MIS and WMIS problems in this common data stream model and we summarize the state of the art in Table 1. Our contributions are as follows:

1. **Unit-length Intervals.** Our main algorithmic contribution is a turnstile streaming algorithm achieving a $(2+\epsilon)$-approximation to α and β in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. We also show a matching lower bound, i.e., any (possibly randomized) algorithm approximating

[1] See `https://en.wikipedia.org/wiki/Interval_scheduling`.

Table 1 The best known upper and lower bounds for estimating α and β in insertion-only and turnstile streams (defined below). Note, the weight and length above are still polynomially bounded in n. The folklore result follows from partitioning the input into $O(\log(n))$ weight classes, estimating α on each one in parallel and taking the maximum estimate.

Problem	Insertion-Only Streams		Turnstile Streams	
	upper bound	lower bound	upper bound	lower bound
Unit Intervals Unit Weight	$3/2+\epsilon$ [10]	$3/2-\epsilon$ [10]	$2+\epsilon$ Thm 8	$2-\epsilon$ Thm 16
Unit Intervals Arbitrary Weight	$3/2+\epsilon$ Thm 22	$3/2-\epsilon$ [10]	$2+\epsilon$ Thm 8	$2-\epsilon$ Thm 16
Unit Disks Arbitrary Weight	$\frac{8\sqrt{3}}{\pi}+\epsilon$ Thm 21	$2-\epsilon$ Thm 25	$\frac{8\sqrt{3}}{\pi}+\epsilon$ Thm 21	$2-\epsilon$ Thm 16

α up to a $(2-\epsilon)$ factor requires $\Omega(n)$ space. Interestingly, this shows a strict separation between insertion-only and turnstile models since [10] show that a $3/2$ approximation is tight in the insertion-only model.

An unintuitive yet crucial message here is that attaching polynomially bounded weights to intervals does not affect the approximation factor. Along the way, we also obtain new algorithms for estimating β in insertion-only streams which are presented in Section C.

2. **Arbitrary Length Intervals.** For arbitrary length intervals, we give a one-pass turnstile streaming algorithm that achieves a $(1+\epsilon)$-approximation to α under the assumption that the degree of the interval intersection graph is bounded by $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. Our algorithm achieves $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. We also study the problem for arbitrary lengths by parameterizing the ratio of the longest to the shortest interval. We give a one-pass turnstile streaming algorithm that achieves a $(2+\epsilon)$-approximation to α, where the space complexity is parameterized by W_{max}, which is an upper bound on the length of an interval assuming the minimum interval length is 1. Here, the space complexity of our algorithm is $\text{poly}\left(W_{max}\frac{\log(n)}{\epsilon}\right)$ and this algorithm gives sublinear space whenever W_{max} is sublinear.
3. **Unit-radius Disks.** We show that we can extend the ideas developed for unit-length intervals in turnstile streams to unit disks in the 2-d plane. We describe an algorithm achieving an $\left(\frac{8\sqrt{3}}{\pi}+\epsilon\right)$-approximation to α and β in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. One key idea in the algorithm is to use the hexagonal circle packing for the plane, where the fraction of area covered is $\frac{\pi}{\sqrt{12}}$ and our approximation constant turns out to be $4\cdot\frac{\sqrt{12}}{\pi}$.
We also show a lower bound that any (possibly randomized) algorithm approximating α or β for disks in insertion-only streams, up to a $(2-\epsilon)$ factor requires $\Omega(n)$ space. This shows a strict separation between estimating intervals and disks in insertion-only streams.

2 Related Work

There has been considerable work on streaming algorithms for graph problems. Well-studied problems include finding sparsifiers, identifying connectivity structure, building spanning trees, and matchings; see the survey by McGregor [46]. Recently, Cormode et. al. [22] provide guarantees for estimating the cardinality of a maximum independent set of general graphs via the Caro-Wei bound. Emek, Halldorsson and Rosen [24] studied estimating the cardinality of the maximum independent set for interval intersection graphs in insertion-only

streams. They output an independent set that is a $\frac{3}{2}$-approximation to the optimal (OPT) for unit-length intervals and a 2-approximation for arbitrary-length intervals in $O(|\mathsf{OPT}|)$ space. Note that $|\mathsf{OPT}|$ could be $\Theta(n)$ which is a prohibitive amount of space.

Subsequently, Cabello and Perez-Lantero [10] studied the problem of estimating the *cardinality* of OPT, which we denote by α, for unit-length and arbitrary length intervals in one-pass insertion-only streams. For unit-length intervals in insertion-only streams, Cabello and Perez-Lantero [10] give a $(\frac{3}{2}+\epsilon)$ approximation to α in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. Additionally, they show that this approximation factor is tight, since any algorithm achieving a $(\frac{3}{2}-\epsilon)$-approximation to α requires $\Omega(n)$ space. For arbitrary-length intervals they give a $(2+\epsilon)$-approximation to α in $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ space. Additionally, they show that the approximation factor is tight, since any algorithm achieving a $(2-\epsilon)$-approximation to α requires $\Omega(n)$ space. Recently, [23] studied MIS of intersection graphs in insertion-only streams. They show achieving a $(5/2-\epsilon)$-approximation to MIS of squares requires $\Omega(n)$ space.

To the best of our knowledge there is no prior work on the problem of Maximum Independent Set of unit disks in turnstile streams. In the offline setting, the first PTAS for MIS of disks was developed by [26] and later improved in running time by Chan [14], while [36] shows a PTAS for MIS of $k \times k$ squares. We note that these algorithms require space linear in the number of disks and use a dynamic programming approach that is not suitable for streaming scenarios.

We note that MIS can also be viewed as a natural generalization of the distinct elements problem that has received considerable attention in the streaming model. This problem was first studied in the seminal work of [29] and a long sequence of work has addressed its space complexity in both insertion-only and turnstile streams [3], [7], [32], [27], [28], [40], [9] and [23].

3 Notation and Problem Definitions

We let $D(d_j, r_j, w_j)$ be a disk in $\mathbb{R}^d$, where $d \in \{1, 2\}$, such that it is centered at a point $d_j \in \mathbb{R}^d$ with radius $r_j \in \mathbb{N}$ and weight w_j. We represent $D(d_j, r_j, w_j)$ using the short form D_j when d_j, r_j and w_j are clear from context. Note, we use the same notation to denote intervals in $d = 1$. For a set $\mathcal{P} \subseteq \mathbb{R}^d$ of n disks (unweighted or weighted), let G be the induced graph formed by assigning a vertex to each disk and adding an edge between two vertices if the corresponding disks intersect. We call G an intersection graph. The Maximum Independent Set (MIS) and Weighted Maximum Independent Set (WMIS) problems in the context of intersection graphs are defined as follows:

▶ **Definition 1** (Maximum Independent Set). *Let $\mathcal{P} = \{D_1, D_2 \ldots, D_n\} \subseteq \mathbb{R}$ be a set of n disks such that each weight $w_j = 1$ for $j \in [n]$. The MIS problem is to find the largest disjoint subset $\mathcal{S}$ of $\mathcal{P}$ (i.e., no two objects in $\mathcal{S}$ intersect). We denote the cardinality of this set by α.*

▶ **Definition 2** (Weighted Maximum Independent Set). *Let $\mathcal{P} = \{D_1, D_2 \ldots, D_n\} \subseteq \mathbb{R}^d$ be a set of n weighted disks. We let the weight $w_{\mathcal{S}}$ of a subset $\mathcal{S} \subseteq \mathcal{P}$ be $w_{\mathcal{S}} = \sum_{D_j \in \mathcal{S}} w_j$. The WMIS Problem is to find a disjoint (i.e., non overlapping) subset $\mathcal{S}$ of $\mathcal{P}$ whose weight $w_{\mathcal{S}}$ is maximum. We denote the weight of the WMIS by β.*

For a set $\mathcal{P}$ of disks, let $\mathsf{OPT}_{\mathcal{P}}$ denote MIS or WMIS of $\mathcal{P}$. We use $|\mathsf{OPT}_{\mathcal{P}}|$ to denote the cardinality of MIS as well as the weight of WMIS for $\mathcal{P}$. When the set $\mathcal{P}$ is clear from context, we omit it. Next, we define the two streaming models we consider. In our context,

an *insertion-only* stream provides sequential access to the input, which is an ordered set of objects such that at any given time step a new interval arrives. *Turnstile streams* are an extension of this model such that at any time step, previously inserted objects can be deleted. An algorithm in the streaming model has access to space sublinear in the size of the input and is restricted to making one pass over the input.

For proving our lower bounds, we work in the two player one-way randomized communication complexity model, where the players are denoted by Alice and Bob, who have private randomness. The input of Alice is denoted by X and the input for Bob is denoted by Y. The objective is for Alice to communicate a message to Bob and compute a function $f : X \times Y \to \{0, 1\}$ on the joint inputs of the players. The communication is one-way and w.l.o.g. Alice sends one message to Bob and Bob outputs a bit denoting the answer to the communication problem. Let $\Pi(X, Y)$ be the random variable that denotes the transcript between sent from Alice to Bob when they execute a protocol Π.

A protocol Π is called a δ-error protocol for function f if there exists a function Π_{out} such that for every input $Pr[\Pi_{out}(\Pi(X, Y)) = f(X, Y)] \geq 1 - \delta$. The communication cost of a protocol, denoted by $|\Pi|$, is the maximum length of $\Pi(X, Y)$ over all possible inputs and random coin flips of the two players. The randomized communication complexity of a function f, $R_\delta(f)$, is the communication cost of the best δ-error protocol for computing f.

4 Technical Overview

In this section, we summarize our results and briefly describe the main technical ideas in our algorithms and lower bounds. We note that our results hold in the recently introduced Sketching Model [49]. This model captures applications of sketches in turnstile streams, distributed computing, communication complexity and property testing. While Sun et. al. study graph problems such as dynamic connectivity and triangle detection, we initiate the study of dynamic Maximum Independent Set in this model. While we state our results in for turnstile streams, they immediately extend to the sketching model.

4.1 Unit-length Intervals

Our main algorithmic contribution is to provide an estimate that obtains a $(2+\epsilon)$-approximation to WMIS of unit-length intervals in turnstile streams :

▶ **Theorem 3** (Theorem 8, informal)**.** *For any $\epsilon > 0$, there exists a turnstile streaming algorithm that outputs an estimate such that with probability at least $99/100$, it is a $(2+\epsilon)$-approximation to WMIS of unit intervals (polynomially bounded weights) and the algorithm requires $poly\left(\frac{\log(n)}{\epsilon}\right)$ space.*

A naïve approximation. We start by describing a simple approach (Algorithm 1) to obtain a 9-approximation. The algorithm proceeds by imposing a grid of side length 1 and shifts it by a random integer. This is a standard technique used in geometric algorithms. We then snap each interval to the cell containing the center of the interval and partition the real line into odd and even cells. This partitions the input space such that intervals landing in distinct odd (even) cells are pairwise independent. Let $\mathcal{C}_e$ be the set of all even cells and $\mathcal{C}_o$ be the set of all odd cells.

By averaging, either $|\mathsf{OPT}_{\mathcal{C}_e}|$ or $|\mathsf{OPT}_{\mathcal{C}_o}|$ is at least $\frac{\mathsf{OPT}}{2}$, where $|\mathsf{OPT}|$ is the max weight independent set of intervals. We develop an estimator that gives a $(1+\epsilon)$-approximation to $|\mathsf{OPT}_{\mathcal{C}_e}|$ as well as $|\mathsf{OPT}_{\mathcal{C}_o}|$. Therefore, taking the max of the two estimators, we obtain a $(2+\epsilon)$-approximation to $|\mathsf{OPT}|$.

Having reduced the problem to estimating $|\mathsf{OPT}_{\mathcal{C}_e}|$, we observe that for each even cell only the max weight interval landing in the cell contributes to $\mathsf{OPT}_{\mathcal{C}_e}$. Then, partitioning the cells in $\mathcal{C}_e$ into poly(log(n)) geometrically increasing weight classes based on the max weight interval in each cell and approximately counting the number of cells in each weight class suffices to estimate $|\mathsf{OPT}_{\mathcal{C}_e}|$ up to a $(1+\epsilon)$-factor.

Algorithm 1 Naïve Approximation.

Input: Given a turnstile stream $\mathcal{P}$ with weighted unit intervals, where the weights are polynomially bounded, ϵ and $\delta > 0$, Naïve Approximation outputs a $(9+\epsilon)$-approximation to β with probability $1-\delta$.

1. Randomly shift a grid Δ of side length 1. Partition the cells into even and odd, denoted by $\mathcal{C}_e$ and $\mathcal{C}_o$.
2. Consider a partition of cells in $\mathcal{C}_e$ into $b = \text{poly}(\log(n))$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+1/2)^i \leq m(c) < (1+1/2)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c (this is not an algorithmic step since we do not know this partition a priori). Create a substream for each weight class $\mathcal{W}_i$ denoted by $\mathcal{W}'_i$.
3. For each new interval $D(d_j, 1, w_j)$, feed it to substream $\mathcal{W}'_i$ if $w_j \in [(1+1/2)^i, (1+1/2)^{i+1})$. For each substream $\mathcal{W}'_i$, maintain a $(1 \pm \epsilon)$-approximate ℓ_0-estimator (described below).
4. Let t_i be the ℓ_0 estimate corresponding to $\mathcal{W}'_i$. Let $X_e = \frac{2}{9(1+\epsilon)} \sum_{i \in [b]} (1+1/2)^{i+1} t_i$.
5. Repeat Steps 2-6 for the odd cells $\mathcal{C}_o$ to obtain the corresponding estimator X_o.

Output: $\max(X_e, X_o)$

Given such a partition, we can approximate the number of cells in each weight class by running an ℓ_0 norm estimator. Estimating the ℓ_0 norm of a vector in *turnstile streams* is a well studied problem and a result of Kane, Nelson and Woodruff [40] obtains a $(1 \pm \epsilon)$-approximation in poly$(\frac{\log(n)}{\epsilon})$ space. However, we do not know the partition of the cells into the weight classes a priori and this partition can vary drastically over the course of a stream given that intervals can be deleted. Therefore, the main technical challenge is to simulate this partition in *turnstile streams*.

As a first attempt, consider a partition of cells in $\mathcal{C}_e$ into $b = \text{poly}(\log(n))$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+1/2)^i \leq m(c) < (1+1/2)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c. Create a substream for each weight class $\mathcal{W}_i$ and feed an input interval into this substream if its weight lies in the range $[(1+1/2)^i, (1+1/2)^{i+1})$. Let t_i be the corresponding ℓ_0 estimate for this substream. Approximate the contribution of $\mathcal{W}_i$ by $(1+1/2)^{i+1} \cdot t_i$. Sum up the estimates for all $i \in [b]$ to obtain an estimate for $|\mathsf{OPT}_{\mathcal{C}_e}|$.

We note that there are two issues with our algorithm. First, we overestimate the weight of intervals in class $\mathcal{W}_i$ by a factor of 3/2 and second, for a given cell we sum up the weights of all intervals landing in it, instead of taking the maximum weight for the cell. In the worst case, we approximate the true weight of a contributing interval, $(3/2)^{i+1}$, with $\sum_{i'=1}^{i} (3/2)^{i'+1} \leq 3((3/2)^{i+1} - 1)$. Note, we again overestimate the weight, this time by a factor of 3. Combined with the approximation for the ℓ_0 norm, we obtain a weaker $(\frac{9}{2} + \epsilon)$-approximation to $|\mathsf{OPT}_{\mathcal{C}_e}|$ in the desired space. From our discussion above, this implies a $(9+\epsilon)$-approximation to $|\mathsf{OPT}|$. We also note that this attempt is not futile as we use the above algorithm as a subroutine subsequently.

A refined attempt. Next, we describe an algorithm that estimates $|\mathsf{OPT}_{\mathcal{C}_e}|$ up to a $(1+\epsilon)$-factor. Here, we use more sophisticated techniques to simulate a finer partition of the cells in $\mathcal{C}_e$ into geometrically increasing weight classes in turnstile streams. One key algorithmic tool

we use here is a streaming algorithm for k-Sparse Recovery: given an input vector x such that x receives coordinate-wise updates in the turnstile streaming model and has at most k non-zero entries at the end of the stream of updates, there exist data structures that exactly recover x at the end of the stream. As mentioned in Berinde et al. [8], the k-tail guarantee is a sufficient condition for k-Sparse Recovery, since in a k-sparse vector, the elements of the tail are 0. We note that the Count-Sketch Algorithm [17] has a k-tail guarantee in *turnstile streams*.

This time around, we consider partitioning cells in $\mathcal{C}_e$ into poly $\left(\epsilon^{-1}\log(n)\right)$ weight classes, creating a substream for each one and computing the corresponding ℓ_0 norm. We also assume we know $|\mathsf{OPT}_{\mathcal{C}_e}|$ up to a constant (this can be simulated in *turnstile streams*). Formally, given $b = \text{poly}\left(\log(n), \epsilon^{-1}\right)$ weight classes, for all $i \in [b]$, let $\mathcal{W}_i$ denote the set of even cells with maximum weight sandwiched in the range $[(1+\epsilon)^i, (1+\epsilon)^{i+1})$. We then simulate sampling from the partition by subsampling cells in each $\mathcal{W}_i$ at the start of the stream, agnostic to the input. We do this at different sampling rates , i.e. for all $i \in [b]$, we subsample the cells in $\mathcal{W}_i$ with probability roughty $(1+\epsilon)^i/|\mathsf{OPT}_{\mathcal{C}_e}|$.

This presents several issues, as we cannot subsample non-empty cells in turnstile streams a priori. Further, if a weight class has a small number of non-empty cells, we cannot recover accurate estimates for the contribution of this weight class to $|\mathsf{OPT}_{\mathcal{C}_e}|$ at any level of the subsampling. To address the first issue, we agnostically sample cells from $\mathcal{C}_e$ according to a carefully chosen range of sampling rates and create a substream for each one. We then run a sparse recovery algorithm on the resulting substreams. At the right subsampling rate, we note that the resulting substream is sparse since we can filter out cells that belong to smaller weight classes. Further, we can ensure that the number of cells that survive from the relevant weight class (and larger classes) is small. Therefore, we recover all such cells using the sparse recovery algorithm.

To address the second issue, we threshold the weight classes that we consider in the algorithm based on the relative fraction of non-empty cells in them. This threshold can be computed in the streaming algorithm using the ℓ_0-norm estimates for each weight class. All the weight classes below the threshold together contribute at most an ϵ-fraction of $|\mathsf{OPT}_{\mathcal{C}_e}|$ and though we cannot achieve concentration for such weight classes, we show that we do not overestimate their contribution. Further, for all the weight classes above the threshold, we can show that sampling at the right rate can recover enough cells to achieve concentration.

We complement the above algorithmic result with a matching lower bound, i.e., a $(2-\epsilon)$-approximation to MIS, for any $\epsilon > 0$, requires $\Omega(n)$ space. This follows from an easy application of the Augmented Indexing problem. We note that our result combined with the 3/2-approximation by [11] implies an unexpected separation between insertion-only and turnstile streams.

4.2 Parametrized Algorithms for Arbitrary Length Intervals

In light of the lower bound discussed above, we identify two sources contributing to the streaming hardness of MIS for arbitrary length intervals : the number of pair-wise intersections (max-degree) and the ratio of the longest to shortest interval (scale). We show that when either of these quantities is poly-logarithmically bounded, we can approximate MIS for arbitrary length intervals.

Instead of assuming the max-degree or scale is bounded, we instead provide algorithms paramterized by these quantities. First, let the number of pair-wise intersections be bounded by $\kappa_{\max}$. Then,

▶ **Theorem 4** (Theorem 17, informal). *For $\epsilon > 0$, there exists a turnstile streaming algorithm that takes as input a set of unit-weight arbitrary-length intervals, with at most $\kappa_{\max}$ pairwise intersections and with probability $99/100$, outputs a $(1+\epsilon)$-approximation to MIS in poly$(\log(n), \epsilon^{-1}, \kappa_{\max})$ space.*

This result requires several new algorithmic ideas. Observe, placing a unit grid no longer suffices since the intervals now span different lengths. Therefore, we impose a nested grid on our input, where the grid size is geometrically increasing, and randomly shift it. Further, observe that the natural strategy that partitions the interval into geometrically increasing *length classes* and estimates each partition up to $1+\epsilon$ does not work since the intervals overlap.

We therefore define the following object that uniquely determines intervals of a particular *length class* contributing to the MIS :

▶ **Definition 5** (r_i-Structure). *We define an r_i-Structure to be a subset of the Nested Grid, such that there exists an interval at the i^{th} grid level, there exist no intervals in the grid at any level $i' > i$ and all the intervals in the grid at levels $i' < i$ intersect the interval at the i^{th} level.*

It is easy to see any interval that contributes to MIS corresponds to an r_i-Structure for some i. Therefore, it suffices to estimate the number of r_i-Structures for all i. Following our approach for unit intervals, we again use k-Sparse Recovery as our main tool. At a high level, we sub-sample poly$(\log(n), \epsilon^{-1})$ r_i-Structures from the set of all such structures at level i, and create a new substream for each i. We then run a $\kappa_{\max}$-Sparse Recovery Algorithm on each substream. We show that at the end of the stream, we obtain an estimate of the number of r_i-Structures at level i that concentrates. Since the structures form a partition, our overall estimate is simply the sum of the estimates obtained for each i.

The main algorithmic challenge here is to show that we can indeed detect and subsample the r_i-structures. These structures are defined in a way that takes into account how many intervals appear in the nested grid both above and below a given interval. Therefore, it is unclear how to track such updates as they constantly change over the stream. However, observe that since our space is parameterized by the max-degree, we can afford to store an r_i-Structure completely in memory.

Given a randomly sampled cell from the i-th level of the nested grid, we assume this cell contributes an r_i-structure. We then run $\kappa_{\max}$-Sparse Recovery on this cell. Our main insight is that at the end of the stream we can verify whether this cell indeed contributed an r_i-structure since we recover the nested intervals *exactly*. The final remaining challenge is to ensure that our sub-sample contains a sufficient number of non-empty structures for each level and the resulting estimate concentrates. We describe these details in Section A.1.

Finally, we show that similar algorithmic ideas also result in a turnstile streaming algorithm, if parametrize the input by the $W_{\max}$, the ratio of the largest to smallest interval :

▶ **Theorem 6** (Theorem 20, informal). *For $\epsilon > 0$, there exists a turnstile streaming algorithm that takes as input a set of unit-weight arbitrary-length intervals, with $W_{\max}$ being an upper bound on the ratio of the largest to smallest interval, and with probability $99/100$, outputs a $(2+\epsilon)$-approximation to MIS in poly$(\log(n), \epsilon^{-1}, W_{\max})$ space.*

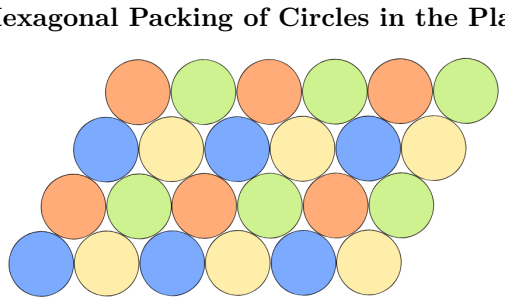


Figure 4.1 We illustrate the hexagonal circle packing in the Euclidean Plane. Each color represents an equivalence class. Observe that input disks that are centered in distinct circles of the same equivalence class are independent, since the circles are at least 2 units apart.

4.3 Unit-Radius Disks

We generalize the WMIS turnstile streaming algorithm for unit length intervals to unit radius disks in $\mathbb{R}^2$. The approximation ratio for disks is closely related to the optimal circle packing constant. We leverage the hexagonal packing of circles in the 2-D plane to obtain the following result:

▶ **Theorem 7** (Theorem 21, informal). *There exists a turnstile streaming algorithm achieving a* $\left(\frac{8\sqrt{3}}{\pi}+\epsilon\right)$*-approximation to estimate WMIS of unit disks with constant probability and in* $poly\left(\frac{\log(n)}{\epsilon}\right)$ *space.*

We note that a greedy algorithm for unweighted disks obtains a 5-approximation to α [25] and the space required is $O(\alpha)$. The greedy algorithm can be extended to obtain a $(5+\epsilon)$-approximation in $\text{poly}\left(\frac{\log n}{\epsilon}\right)$ space using the sampling approach we presented in Section 5. However, beating the approximation ratio achieved by the greedy algorithm requires geometric insight. Critically, we use the hexagonal packing of unit circles in a plane introduced by Lagrange , which was shown to be optimal by Toth [16].

The hexagonal packing covers a $\frac{\pi}{\sqrt{12}}$ fraction of the area in two dimensions. We then partition the unit circles in the hexagonal packing into equivalence classes such that two circles in the same equivalence class are at least a unit distance apart. Formally, let c_1, c_2 be two unit circles in the hexagonal packing of the plane lying in the same equivalence class. Then, for all points $p_1 \in c_1$, $p_i \in c_2$, $\|p_1 - p_2\|_2 \geq 1$. Therefore, if two input disks of unit radius have centers lying in distinct circles belong to the same equivalence class, the disks must be independent, as long as the disk are not centered on the boundary of the circles.

Randomly shifting the underlying hexagonal packing ensures this happens with probability 1. We then show that we can partition the hexagonal packing into four equivalence classes such that their union covers all the circles in the packing and disks lying in distinct circles of the same equivalence class are independent.

Algorithmically, we first impose a grid Δ of the hexagonal packing of circles with radius 1 and shift it by a random integer. We discard all disks that do not have centers lying inside the grid Δ. Given that a hexagonal packing covers a $\pi/\sqrt{12}$ fraction of the area, in expectation, we discard a $(1-\pi/\sqrt{12})$ fraction of $|\mathsf{OPT}|$. We note that if we could accurately estimate the remaining WMIS, and scale the estimator by $\sqrt{12}/\pi$, we would obtain a $(\sqrt{12}/\pi)$-approximation to $|\mathsf{OPT}|$. Let $||\mathsf{OPT}|_{\text{hp}}|$ denote the remaining WMIS. By Theorem 16 such an approximation requires $\Omega(n)$ space.

We then observe that the hexagonal circle packing grid can be partitioned into four equivalence classes. We use $\mathcal{C}_1, \mathcal{C}_2, \mathcal{C}_3$ and $\mathcal{C}_4$ to denote these equivalence classes. Since the equivalence classes form a partition of the hexagonal packing, at least one of them must contain a 1/4-fraction of the remaining maximum independent set. W.l.o.g, let $\mathcal{C}_1$ be the partition that contributes the most to $|\mathsf{OPT}|$. Then, $|\mathsf{OPT}_{\mathcal{C}_1}| \geq \frac{1}{4}|\mathsf{OPT}_{\text{hp}}|$. Therefore, we focus on designing an estimator for $\mathcal{C}_1$. We show a $(1+\epsilon)$-approximation to $\mathcal{C}_1$ in poly $\left(\log(n), \epsilon^{-1}\right)$ space generalizing the algorithmic ideas we introduced for Theorem 8. This implies an overall $\left(\frac{4\sqrt{12}}{\pi} + \epsilon\right) = \left(\frac{8\sqrt{3}}{\pi} + \epsilon\right)$ approximation for $|\mathsf{OPT}|$.

5 Weighted Interval Selection for Unit Intervals

In this section, we present an algorithm to approximate the weight of the maximum independent set, β, for unit-length intervals in turnstile streams. Interestingly, we note that estimating β has the same complexity as approximating α for unit-length intervals. That is, we obtain a $(2+\epsilon)$-approximation to β in the turnstile model, which immediately implies $(2+\epsilon)$-approximation for α, where the weights are identical. We complement this result with a lower bound that shows any $(2-\epsilon)$-approximation to α requires $\Omega(n)$ space. The main algorithmic guarantee we achieve is as follows:

▶ **Theorem 8.** *Let $\mathcal{P}$ be a turnstile stream of weighted unit intervals such that the weights are polynomially bounded in n and let $\epsilon \in (0, 1/2)$. There exists an algorithm that outputs an estimator Y such that with probability at least $9/10$ the following guarantees hold:*

1. $\frac{\beta}{2(1+\epsilon)} \leq Y \leq \beta$.
2. *The total space used is poly* $\left(\frac{\log(n)}{\epsilon}\right)$.

We first impose a grid Δ of side length 1 and shift it by a random integer. We then snap each interval to the cell containing the center of the interval and partition the real line into odd and even cells. Let $\mathcal{C}_e$ be the set of all even cells and $\mathcal{C}_o$ be the set of all odd cells. By averaging, either $|\mathtt{OPT}_{\mathcal{C}_e}|$ or $|\mathtt{OPT}_{\mathcal{C}_o}|$ is at least $\frac{\beta}{2}$. We describe an estimator that gives a $(1+\epsilon)$-approximation to $|\mathtt{OPT}_{\mathcal{C}_e}|$ and $|\mathtt{OPT}_{\mathcal{C}_o}|$. W.l.o.g let $|\mathtt{OPT}_{\mathcal{C}_e}| \geq |\mathtt{OPT}_{\mathcal{C}_o}|$. Therefore, taking the max of the two estimators, we obtain a $(2+\epsilon)$-approximation to β.

Having reduced the problem to estimating $|\mathtt{OPT}_{\mathcal{C}_e}|$, we observe that each even cell has at most 1 interval, namely the max weight interval landing in the cell, contributing to $\mathtt{OPT}_{\mathcal{C}_e}$. Then, partitioning the cells in $\mathcal{C}_e$ into poly$(\log(n))$ weight classes based on the max weight interval in each cell and approximately counting the number of cells in each weight class suffices to estimate $|\mathtt{OPT}_{\mathcal{C}_e}|$ up to a $(1+\epsilon)$-factor. Given such a partition, we can create a substream for each weight class in the partition and compute the ℓ_0 norm of each substream. However, we do not know the partition of the cells into the weight classes a priori and this partition can vary drastically over the course of stream given that intervals can be deleted. The main technical challenge is to simulate this partition. A key tool we use is to estimate the ℓ_0 norm of a vector in turnstile streams. Kane, Nelson and Woodruff [40] showed how to obtain a $(1 \pm \epsilon)$-approximation to the ℓ_0-norm of a vector in poly$(\frac{\log(n)}{\epsilon})$ space.

▶ **Theorem 9** (ℓ_0-Norm Estimation [40])**.** *In the turnstile model, there is an algorithm for $(1 \pm \epsilon)$-approximating the ℓ_0-norm (number of non-zero coordinates) of a vector using space poly* $\left(\frac{\log(n)}{\epsilon}\right)$ *with success probability $2/3$.*

We begin by describing a simple algorithm which obtains a weaker $(9/2+\epsilon)$-approximation to $|\mathtt{OPT}_{\mathcal{C}_e}|$ and in turn a $(9+\epsilon)$-approximation to β. Formally, consider a partition of cells in $\mathcal{C}_e$ into $b = \text{poly}(\log(n))$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+1/2)^i \leq m(c) < (1+1/2)^{i+1}\}$,

where $m(c)$ is the maximum weight of an interval in c. Create a substream for each weight class $\mathcal{W}_i$, denoted by $\mathcal{W}'_i$, and feed an input interval into this substream if its weight lies in the range $[(1+1/2)^i, (1+1/2)^{i+1})$. Let t_i be the corresponding ℓ_0 estimate for substream $\mathcal{W}'_i$. Then, we can approximate the contribution of $\mathcal{W}_i$ by $(1+1/2)^{i+1} \cdot t_i$. Summing over the b weight classes gives an estimate for $|\texttt{OPT}_{\mathcal{C}_e}|$.

Given access to an algorithm for estimating the ℓ_0-norm, the Naïve Approximation Algorithm (1) satisfies the following guarantee:

▶ **Lemma 10.** *The Naïve Approximation Algorithm (1) outputs an estimate X such that with probability $99/100$, $\frac{\beta}{9(1+\epsilon)} \leq X \leq \beta$ and runs in space poly$\left(\frac{\log(n}{\epsilon}\right)$.*

Proof. We observe that for each non-empty cell $c \in \mathcal{C}_e$, there is exactly 1 interval that can contribute to $|\texttt{OPT}_{\mathcal{C}_e}|$ since each cell of the grid has side length 1 and all intervals falling in a given cell pairwise intersect. This contributing interval lies in some weight class $\mathcal{W}_i$ and our estimator approximates its weight as $(1+1/2)^{i+1}$. Here, the weights of the intervals are sandwiched between $(1+1/2)^i$ and $(1+1/2)^{i+1}$. Therefore, we overestimate the weight by a factor of at most $3/2$.

Further, instead of taking the maximum over each cell c, we may have inserted intervals that lie in c into all substreams $\mathcal{W}'_i$. Therefore, we take the sum of our geometrically increasing weight classes over that cell. In the worst case, we approximate the true weight of a contributing interval, $(3/2)^{i+1}$, with $\sum_{i'=1}^{i}(3/2)^{i'+1} = 3((3/2)^{i+1} - 1)$. Note, we again overestimate the weight, this time by a factor of 3.

Finally, Theorem 9 overestimates the ℓ_0-norm of $\mathcal{W}_i$ by at most $1+\epsilon$ with probability at least $2/3$. We boost this probability by running $O(\log(n))$ estimators and taking the median. Union bounding over all $i \in [b]$, we simultaneously overestimate the ℓ_0-norm of all $\mathcal{W}_i$ by at most $1+\epsilon$ with probability at least $99/100$. Therefore, the overall estimator is a $(9/2+\epsilon)$-approximation to $|\texttt{OPT}_{\mathcal{C}_e}|$. Rescaling our estimator by the above constant underestimates $|\texttt{OPT}_{\mathcal{C}_e}|$. Finally, $|\texttt{OPT}_{\mathcal{C}_e}| \geq \beta/2$ and $\frac{\beta}{(9+\epsilon)} \leq X \leq \beta$.

Since our weights are polynomially bounded, we create poly$\left(\log_{1+\epsilon}(n)\right)$ substreams and run an ℓ_0 estimator from Theorem 9 on each substream. Therefore, the total space used by Algorithm 1 is poly$\left(\frac{\log(n}{\epsilon}\right)$. ◀

We can thus assume we know β and $|\texttt{OPT}_{\mathcal{C}_e}|$ up to a constant by initially making $O(\log(n))$ guesses and running the Naïve Approximation Algorithm for each guess in parallel. At the end of the stream, we know the correct guess up to a constant factor, and thus can output the estimator corresponding to that branch of computation. A key tool we use in this algorithm is k-`Sparse Recovery`. As mentioned in Berinde et al. [8], the k-tail guarantee is a sufficient condition for k-`Sparse Recovery`, since in a k-sparse vector, the elements of the tail are 0. We note that the `Count-Sketch` Algorithm [17] has a k-tail guarantee in turnstile streams.

▶ **Definition 11** (k-`Sparse Recovery`)**.** *Let x be the input vector such that x is updated coordinate-wise in the turnstile streaming model. Then, x is k-sparse if x has at most k non-zero entries at the end of the stream of updates. Given that x is k-sparse, a data structure that exactly recovers x at the end of the stream is referred to as a k-`Sparse Recovery` data structure.*

Intuitively, we again simulate partitioning cells in $\mathcal{C}_e$ into poly $\left(\frac{\log(n)}{\epsilon}\right)$ weight classes according to the maximum weight occurring in each cell. Since we do not know this partition a priori, we initially create $b = O\left(\frac{\log(n)}{\epsilon}\right)$ substreams, one for each weight class and run the

ℓ_0-estimator on each one. We then make $O\left(\frac{\log(n)}{\epsilon}\right)$ guesses for $|\texttt{OPT}_{\mathcal{C}_e}|$ and run the rest of the algorithm for each branch in parallel. Additionally, we run the Naïve Approximation Algorithm to compute the right value of $|\texttt{OPT}_{\mathcal{C}_e}|$ up to a constant factor, which runs in space poly $\left(\frac{\log(n)}{\epsilon}\right)$. Then, we create $b = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ substreams by agnostically sampling cells with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$, where X is the right guess for $|\texttt{OPT}_{\mathcal{C}_e}|$. Sampling at this rate preserves a sufficient number of cells from weight class $\mathcal{W}_i$. We then run a sparse recovery algorithm on the resulting substreams.

We note that the resulting substreams are sparse. To see this, note we can filter out cells that belong weight classes $\mathcal{W}_{i'}$ for $i' < i$ by simply checking if the maximum interval seen so far lies in weight classes $\mathcal{W}_i$ and higher. Further, sampling with probability proportional to $\Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 |\texttt{OPT}_{\mathcal{C}_e}|}\right)$ ensures that the number of cells that survive from weight classes $\mathcal{W}_i$ and above are small. Therefore, we recover all such cells using the sparse recovery algorithm. Note, we limit the algorithm to considering weight classes that have a non-trivial contribution to $\texttt{OPT}_{\mathcal{C}_e}$.

Using the ℓ_0 norm estimates computed above, we can determine the number of non-empty cells in each of the weight classes. Thus, we create a threshold for weight classes that contribute, such that all the weight classes below the threshold together contribute at most an ϵ-fraction of $|\texttt{OPT}_{\mathcal{C}_e}|$ and we can set their corresponding estimators to 0. Further, for all the weight classes above the threshold, we can show that sampling at the right rate leads to recovering enough cells to achieve concentration in estimating their contribution.

Next, we show that the total space used by Algorithm 2 is poly $\left(\frac{\log(n)}{\epsilon}\right)$. We initially create $b = O\left(\frac{\log(n)}{\epsilon}\right)$ substreams, one for each weight class and run an ℓ_0-estimator on each one. Recall, this requires poly $\left(\frac{\log(n)}{\epsilon}\right)$. We then make $O\left(\frac{\log(n)}{\epsilon}\right)$ guesses for $|\texttt{OPT}_{\mathcal{C}_e}|$ and run the rest of the algorithm for each branch in parallel. Additionally, we run Algorithm 1 to compute the right value of $|\texttt{OPT}_{\mathcal{C}_e}|$ up to a constant factor, which runs in space poly $\left(\frac{\log(n)}{\epsilon}\right)$. Then, we create b substreams by sampling cells with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$, for $i \in [b]$. Subsequently, we run a poly $\left(\frac{\log(n)}{\epsilon}\right)$-sparse recovery algorithm on each one. Note, if each sample is not too large, this can be done in poly $\left(\frac{\log(n)}{\epsilon}\right)$ space. Therefore, it remains to show that each sample $\mathcal{S}_i$ is small.

▶ **Lemma 12.** *Given a turnstile stream $\mathcal{P}$, with probability at least 99/100, the Weighted Unit Interval Turnstile Sampling procedure (Algorithm 2) samples poly $\left(\frac{\log(n)}{\epsilon}\right)$ cells from the grid Δ.*

Proof. For $i \in [b]$, let $\mathcal{S}_i$ be a substream of cells in $\mathcal{C}_e$, sampled with probability p_i and having an interval with weight at least $(1+\epsilon)^i$ since we filter out all cells with smaller weight. Then, by an averaging argument, the total number of cells with an interval of weight at least $(1+\epsilon)^i$ is at most $\frac{\beta}{(1+\epsilon)^i}$. Sampling with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$, the expected number of cells from $\mathcal{W}_i$ that survive in $\mathcal{S}_i$ is at most $p_i \frac{\beta}{(1+\epsilon)^i} = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ in expectation. Next, we show that they are never much larger than their expectation. Let X_c be the indicator random variable for cell $c \in \mathcal{W}_i$ to be sampled in $\mathcal{S}_i$ and let μ be the expected number of cells in $\mathcal{S}_i$. By Chernoff bounds,

$$\Pr\left[\sum_c X_c \geq (1+\epsilon)\mu\right] \leq \exp\left(-\frac{2\epsilon^2 \text{poly}(\log(n))}{\text{poly}(\epsilon)}\right) \leq 1/n^k$$

for some large constant k. A similar argument holds for the number of cells from weight class $\mathcal{W}_{i'}$, for $i' > i$, surviving in substream $\mathcal{S}_i$. Note, for all $i' < i$, we never include such a cell from weight class $\mathcal{W}_{i'}$ in our sample $\mathcal{S}_i$, since the filtering step rejects all cells that do not contain an interval of weight at least $(1+\epsilon)^i$. Union bounding over the events that cells $c \in \mathcal{W}_{i'}$ get sampled in $\mathcal{S}_i$, for $i' \geq i$, the cardinality of $\mathcal{S}_i$ is at most poly $\left(\frac{\log(n)}{\epsilon}\right)$ with probability at least $1 - 1/n^{k'}$ for an appropriate constant k'. Since we create b such substreams for $\mathcal{C}_e$, we can union bound over such events in each of them and thus $\bigcup_{i\in[b]} |\mathcal{S}_i|$ is at most poly $\left(\frac{\log(n)}{\epsilon}\right)$ with probability at least 99/100. Since $|\mathcal{C}_e|$ is $|\Delta|/2$, the same result holds for the total cells sampled from Δ. Therefore, the overall space used by Algorithm 2 is poly $\left(\frac{\log(n)}{\epsilon}\right)$. ◀

Algorithm 2 Weighted Unit Interval Turnstile Sampling.

Input: Given a turnstile stream $\mathcal{P}$ with weighted unit intervals, where the weights are polynomially bounded, ϵ and $\delta > 0$, the sampling procedure outputs a $(2+\epsilon)$-approximation to β.

1. Randomly shift a grid Δ of side length 1. Partition the cells into $\mathcal{C}_e$ and $\mathcal{C}_o$.
2. For cells in $\mathcal{C}_e$, snap each interval in the input to a cell c that contains its center. Consider a partitioning of the cells in $\mathcal{C}_e$ into $b =$ poly $\left(\frac{\log(n)}{\epsilon}\right)$ weight classes $\mathcal{W}_i = \{c \in \mathcal{C}_e | (1+\epsilon)^i \leq m(c) \leq (1+\epsilon)^{i+1}\}$, where $m(c)$ is the maximum weight of an interval in c (we do not know this partition a priori.) Create a substream for each weight class $\mathcal{W}_i$ denoted by $\mathcal{W}'_i$.
3. Feed interval $D(d_j, 1, w_j)$ along substream $\mathcal{W}'_i$ such that $w_j \in [(1+\epsilon)^i, (1+\epsilon)^{i+1})$. Maintain a $(1 \pm \epsilon)$-approximate ℓ_0-estimator for each substream. Let $|\mathcal{W}'_i|$ denote the number of non-empty cells in substream $\mathcal{W}'_i$ and $X_{\mathcal{W}'_i}$ be the corresponding estimate returned by the ℓ_0-estimator.
4. Create $O(\log(n))$ substreams, one for each guess of $|\texttt{OPT}_{\mathcal{C}_e}|$. Let X be the guess for the current branch of the computation. In parallel, run Algorithm 1 estimates $|\texttt{OPT}_{\mathcal{C}_e}|$ up to a constant factor. Therefore, at the end of the stream, we know a constant factor approximation to the correct value of $|\texttt{OPT}_{\mathcal{C}_e}|$ and use the estimator from the corresponding branch of the computation.
5. In parallel, for $i \in [b]$, create substream $\mathcal{S}_i$ by subsampling cells in $\mathcal{C}_e$ with probability $p_i = \Theta\left(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}\right)$. Note, this sampling is done agnostically at the start of the stream.
6. Run a poly$\left(\frac{\log(n)}{\epsilon}\right)$-sparse recovery algorithm on each substream $\mathcal{S}_i$. For substream $\mathcal{S}_i$, filter out cells c such that $m(c) < (1+\epsilon)^i$. Let $\mathcal{S}'_i$ be the set of cells recovered by the sparse recovery algorithm. Let $\mathcal{S}'_{i|\mathcal{W}_i}$ be the cells in $\mathcal{S}'_i$ that belong to weight class $\mathcal{W}_i$.
7. Let $X_{\mathcal{W}'} = \sum_{i\in[b]} X_{\mathcal{W}'_i}$. Let Z_c be a random variable such that $Z_c = \frac{(1+\epsilon)^{i+1}}{p_i}$ if $c \in \mathcal{S}'_{i|\mathcal{W}_i}$ and 0 otherwise. If $X_{\mathcal{W}'_i} \geq \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1}b}$, set the estimator for the i^{th} subsample, $Y_i = \sum_{c\in\mathcal{S}'_{i|\mathcal{W}_i}} X_{\mathcal{W}'_i} Z_c / \mathcal{S}'_{i|\mathcal{W}_i}$. Otherwise, set $Y_i = 0$. Let $Y_e = \sum_i Y_i$.
8. Repeat Steps 2-7 for the set $\mathcal{C}_o$ and let Y_o be the corresponding estimator.

Output: $Y = \max(Y_e, Y_o)$.

Next, we show that the estimate returned by our sampling procedure is indeed a $(2+\epsilon)$-approximation. We observe that the union of the $\mathcal{W}_i$'s form a partition of $\mathcal{C}_e$. Therefore, it suffices to show that we obtain a $(1+\epsilon)$-approximation to the WIS for each $\mathcal{W}_i$ with

good probability. Let c denote a cell in $\mathcal{W}_i$ and $\texttt{OPT}_c$ denote the $\texttt{WIS}$ in cell c. We create a substream for each weight class $\mathcal{W}_i$ denoted by $\mathcal{W}'_i$ and let $X_{\mathcal{W}'_i}$ be the corresponding estimate returned by the ℓ_0 norm of $\mathcal{W}'_i$. Let $X_{\mathcal{W}'} = \sum_{i\in[b]} X_{\mathcal{W}'_i}$ denote the sum of the estimates across the b substreams.

We say that weight class $\mathcal{W}_i$ contributes if $X_{\mathcal{W}'_i} \geq \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1}b}$. Note, if we discard all the weight classes that do not contribute we lose at most an ϵ-fraction of β (as shown below). Therefore, setting the estimators corresponding to classes that do not contribute to 0 suffices. The main technical hurdle remaining is to show that if a weight class contributes we can accurately estimate $|\texttt{OPT}_{\mathcal{W}_i}|$.

▶ **Lemma 13.** *Let $Y_e = \sum_i Y_i$ be the estimator returned by Algorithm 2 for the set $\mathcal{C}_e$. Then, $Y_e = (1 \pm \epsilon)|\mathit{OPT}_{\mathcal{C}_e}|$ with probability at least 99/100.*

Proof. We first consider the case when $\mathcal{W}_i$ contributes, i.e., $X_{\mathcal{W}'_i} \geq \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1}b}$. Note, $X_{\mathcal{W}'} = \sum_{i\in[b]} X_{\mathcal{W}'_i}$ is a $(1\pm\epsilon)$-approximation to the number of non-empty cells in $\mathcal{W}$ with probability at least $1-n^{-k}$, where $\mathcal{W} = \bigcup_{i\in[b]} \mathcal{W}_i$, since the ℓ_0-estimator is a $(1\pm\epsilon)$-approximation to the number of non-empty cells in $\mathcal{W}_i$ simultaneously for all i with high probability and the $\mathcal{W}_i$'s are disjoint. Recall, X is the correct guess for $|\texttt{OPT}_{\mathcal{C}_e}|$. Therefore, $(1+\epsilon)^i X_{\mathcal{W}'_i} = \Omega\left(\frac{\epsilon X}{(1+\epsilon)b}\right)$. Then, sampling at a rate $p_i = \Theta(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X})$ implies at least $\Omega\left(\frac{\epsilon X}{(1+\epsilon)^{i+1}b}\right) \cdot \Theta(\frac{b(1+\epsilon)^i \log(n)}{\epsilon^3 X}) = \Omega\left(\frac{\log(n)}{(1+\epsilon)\epsilon^2}\right)$ cells from $\mathcal{W}_i$ survive in expectation. Let X_c denote an indicator random variable for cell $c \in \mathcal{W}_i$ being in substream $\mathcal{S}_i$. Then, by a Chernoff bound,

$$\Pr\left[\sum_{c\in\mathcal{W}_i} X_c \leq (1-\epsilon)\left(\frac{\log(n^{-c})}{2\epsilon^2}\right)\right] \leq \exp\left(\frac{-2\epsilon^2 \log(n^{-c})}{2\epsilon^2}\right) \leq n^{-c}$$

for some constant c. Union bounding over all the random events similar to the one above for $i \in [b]$, simultaneously for all i, the number of cells from $\mathcal{W}_i$ in $\mathcal{S}_i$ is at least $\Omega\left(\frac{\log(n)}{\epsilon^2}\right)$ with probability at least $1 - 1/n^k$ for some constant k. Note, for $i' < i$, no cell $c \in \mathcal{W}_{i'}$ exists in $\mathcal{S}_i$ since the filter step removes all cells c such that $m(c) < (1+\epsilon)^i$.

Next, consider a weight class $\mathcal{W}_{i'}$ for $i' > i$ such that it contributes. We upper bound the number of cells from $\mathcal{W}_{i'}$ that survive in substream $\mathcal{S}_i$. Note, weight class $\mathcal{W}_{i'}$ contains at most $\frac{\beta}{(1+\epsilon)^{i+1}}$ non empty cells for $i' > i$. In expectation, at most $\frac{\beta}{(1+\epsilon)^{i+1}} \cdot p_i = O\left(b\frac{\log(n)}{(1+\epsilon)\epsilon^3}\right)$ cells from $\mathcal{W}_{i'}$ survive in sample $\mathcal{S}_i$, for $i' > i$. By a Chernoff bound, similar to the one above, simultaneously for all $i' > i$, at most $O\left(b\frac{\log(n)}{(1+\epsilon)\epsilon^3}\right)$ cells from $\mathcal{W}_{i'}$ survive, with probability at least $1 - 1/n^{k'}$.

Now, we observe that the total number of cells that survive the sampling process in substream $\mathcal{S}_i$ is poly $\left(\frac{\log(n)}{\epsilon}\right)$ and therefore, they can be recovered exactly by the poly $\left(\frac{\log(n)}{\epsilon}\right)$-sparse recovery algorithm. Let the resulting set be denoted by $\mathcal{S}'_i$. We can also compute the number of cells that belong to weight class $\mathcal{W}_i$ that are recovered in the set $\mathcal{S}'_i$ and we denote this by $|\mathcal{S}'_{i|\mathcal{W}_i}|$. Recall, the corresponding estimator is $Y_i = \sum_{c\in\mathcal{S}'_{i|\mathcal{W}_i}} \frac{X_{\mathcal{W}'_i} Z_c}{|\mathcal{S}'_{i|\mathcal{W}_i}|}$, where $Z_c = \frac{(1+\epsilon)^{i+1}}{p_i}$ if $c \in \mathcal{S}'_{i|\mathcal{W}_i}$ and 0 otherwise. We first show we obtain a good estimator for $|\texttt{OPT}_{\mathcal{W}_i}|$ in expectation: $\mathbf{E}[Y_i] = \mathbf{E}\sum_{c\in\mathcal{S}'_{i|\mathcal{W}_i}} X_{\mathcal{W}'_i} Z_c / \mathcal{S}'_{i|\mathcal{W}_i} = (1 \pm 4\epsilon)|\texttt{OPT}_{\mathcal{W}_i}|$.

Since we know that $|\mathcal{S}'_{i|\mathcal{W}_i}| = \Omega(\frac{\log(n)}{(1+\epsilon)\epsilon^2})$, we show that our estimator concentrates. Note, $\mathbf{E}[Y_i] = (1+\epsilon)^{i+1} X_{\mathcal{W}'_i} = \Omega(\frac{\epsilon X}{\log(n)})$. Further, $0 \leq Z_c \leq \frac{(1+\epsilon)^{i+1}}{p_i} = O\left(\frac{(1+\epsilon)^{i+1}\epsilon^3 X}{b(1+\epsilon)^i \log(n)}\right)$. By a Hoeffding bound, $\Pr\left[|Y_i - E[Y_i]| \geq \epsilon E[Y_i]\right] \leq 2\exp\left(\frac{\Omega(\log(n))}{1+\epsilon}\right) \leq 1/n^k$ for some constant

k. Therefore, union bounding over all i, Y_i is a $(1 \pm \epsilon)^2$-approximation to $|\texttt{OPT}_{\mathcal{W}_i}|$ with probability at least $1 - 1/n$. Therefore, if $\mathcal{W}_i$ contributes we obtain a $(1 \pm \epsilon)$-approximation to $|\texttt{OPT}_{\mathcal{W}_i}|$.

In the case where $\mathcal{W}_i$ does not contribute, we set the corresponding estimator to 0. Note, $X_{\mathcal{W}_i'} < \frac{\epsilon X_{\mathcal{W}'}}{(1+\epsilon)^{i+1}b} = \frac{\epsilon(1\pm\epsilon)|\texttt{OPT}_{\mathcal{W}}|}{b} = O(\frac{\epsilon\beta}{b})$. Note, since there are at most b weight classes, discarding all weight classes that do not contribute discards at most $O(\epsilon\beta)$. We therefore lose at most an ϵ-fraction of β by setting the Y_i corresponding to non-contributing weight classes to 0. ◀

Combining Lemmas 10 and 13 finishes the proof for Theorem 8.

5.1 Lower bound for Unit Intervals

Here, we describe a communication complexity lower bound for estimating α for unit-length interval in turnstile streams and thus show the optimality of Theorem 8. Our starting point is the `Augmented Index` problem and its communication complexity is well understood in the two-player one-way communication model. In this model, we have two players Alice and Bob who are required to compute a function based on their joint input and Alice is allowed to send messages to Bob that are a function of her input and finally Bob announces the answer. Note, Bob isn't allowed to send messages to Alice.

▶ **Definition 14** (`Augmented Indexing`). *Let $\texttt{AI}_{n,j}$ denote the communication problem where Alice receives as input $x \in \{0,1\}^n$ and Bob receives an index $j \in [n]$, along with the $x_{j'}$ for $j' > j$. The objective is for Bob to output x_j in the one-way communication model.*

▶ **Theorem 15** (Communication Complexity of $\texttt{AI}_{n,j}$, [47]). *The randomized one-way communication complexity of $\texttt{AI}_{n,j}$ with error probability at most $1/3$ is $\Omega(n)$.*

Let `Alg` be a one-pass turnstile streaming algorithm that estimates α. We show that `Alg` can be used as a subroutine to solve $\texttt{AI}_{n,j}$, in turn implying a lower bound on the space complexity of `Alg`. We formalize this idea in the following theorem:

▶ **Theorem 16.** *Any randomized one-pass turnstile streaming algorithm `Alg` which approximates α to within a $(2-\epsilon)$-factor, for any $\epsilon > 0$, for unit intervals, with at least constant probability, requires $\Omega(n)$ space.*

Proof. Given her input x, Alice constructs a stream of unit-length intervals and runs `Alg` on the stream. For $i \in [n]$, Alice inserts the interval $\left[\frac{2i-x_i}{n^2}, (\frac{2i-x_i}{n^2}) + 1\right]$. She then communicates the state of `Alg` to Bob. Bob uses the message received from Alice as the initial state of the algorithm and continues the stream. Since Bob's input includes an index j and x_i for all $i > j$, Bob deletes all intervals corresponding to such i. Bob then inserts $\left[(\frac{2j-0.5}{n^2}) - 1, \frac{2j-0.5}{n^2}\right]$.

Let us consider the case where $x_j = 1$. We first note that Bob's interval is the leftmost interval in the remaining set. The right endpoint of this interval is $\frac{2j-0.5}{n^2}$. Next, the rightmost interval corresponds to the j^{th} interval inserted by Alice. The left endpoint of this interval is $\frac{2j-1}{n^2}$. Clearly, these intervals intersect each other and intersect all the intervals between them. Therefore, $\alpha = 1$.

Let us now consider the case where $x_j = 0$. Again, Bob's interval is the leftmost with its right endpoint at $\frac{2j-0.5}{n^2}$. However, the left endpoint of Alice's rightmost interval is $\frac{2j}{n^2}$ and thus these two intervals are independent. Therefore, $\alpha \geq 2$. Observe, any $(2-\epsilon)$-approximate algorithm can distinguish between these two cases and solve $\texttt{AI}_{n,j}$. By Theorem 15, any such algorithm requires $\Omega(n)$ communication and in turn $\Omega(n)$ space. ◀

References

1 Pankaj K Agarwal, Marc Van Kreveld, and Subhash Suri. Label placement by maximum independent set in rectangles. *Computational Geometry*, 11(3-4):209–218, 1998.

2 Yuqing Ai, Wei Hu, Yi Li, and David P. Woodruff. New characterizations in turnstile streams with applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016. doi:10.4230/LIPIcs.CCC.2016.20.

3 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. ACM, 1996.

4 Baruch Awerbuch, Yair Bartal, Amos Fiat, and Adi Rosén. Competitive non-preemptive call control. In *SODA*, volume 94, pages 312–320, 1994.

5 Yossi Azar and Oren Gilon. Buffer management for packets with processing times. In *Algorithms-ESA 2015*, pages 47–58. Springer, 2015.

6 Unnar Th Bachmann, Magnús M Halldórsson, and Hadas Shachnai. Online scheduling intervals and t-intervals. *Full version*, 2010.

7 Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, pages 1–10. Springer, 2002.

8 Radu Berinde, Graham Cormode, Piotr Indyk, and Martin J. Strauss. Space-optimal heavy hitters with strong error bounds. In *Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA*, pages 157–166, 2009. doi:10.1145/1559795.1559819.

9 Jarosław Błasiok. Optimal streaming and tracking distinct elements with high probability. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2432–2448. SIAM, 2018.

10 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. In *Algorithms and Data Structures - 14th International Symposium, WADS 2015, Victoria, BC, Canada, August 5-7, 2015. Proceedings*, pages 127–139, 2015. doi:10.1007/978-3-319-21840-3_11.

11 Sergio Cabello and Pablo Pérez-Lantero. Interval selection in the streaming model. *Theoretical Computer Science*, 702:77–96, 2017.

12 Ran Canetti and Sandy Irani. Bounding the power of preemption in randomized scheduling. *SIAM Journal on Computing*, 27(4):993–1015, 1998.

13 Parinya Chalermsook and Julia Chuzhoy. Maximum independent set of rectangles. In *Proceedings of the twentieth annual ACM-SIAM symposium on Discrete algorithms*, pages 892–901. Society for Industrial and Applied Mathematics, 2009.

14 Timothy M Chan. Polynomial-time approximation schemes for packing and piercing fat objects. *Journal of Algorithms*, 46(2):178–189, 2003.

15 Timothy M. Chan and Sariel Har-Peled. Approximation algorithms for maximum independent set of pseudo-disks. *Discrete & Computational Geometry*, 48(2):373–392, 2012. doi:10.1007/s00454-012-9417-5.

16 Hai-Chau Chang and Lih-Chung Wang. A simple proof of thue's theorem on circle packing. *arXiv preprint*, 2010. arXiv:1009.4322.

17 Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, Malaga, Spain, July 8-13, 2002, Proceedings*, pages 693–703, 2002. doi:10.1007/3-540-45465-9_59.

18 Julia Chuzhoy and Alina Ene. On approximating maximum independent set of rectangles. In *Foundations of Computer Science (FOCS), 2016 IEEE 57th Annual Symposium on*, pages 820–829. IEEE, 2016.

19 Brent N Clark, Charles J Colbourn, and David S Johnson. Unit disk graphs. *Discrete mathematics*, 86(1-3):165–177, 1990.

20 Peter Clifford and Ioana Cosma. A simple sketching algorithm for entropy estimation over streaming data. In *Artificial Intelligence and Statistics*, pages 196–206, 2013.

21 Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

22 Graham Cormode, Jacques Dark, and Christian Konrad. Independent set size approximation in graph streams. *CoRR*, abs/1702.08299, 2017. `arXiv:1702.08299`.

23 Graham Cormode, Jacques Dark, and Christian Konrad. Independent sets in vertex-arrival streams. *arXiv preprint*, 2018. `arXiv:1807.08331`.

24 Yuval Emek, Magnús M. Halldórsson, and Adi Rosén. Space-constrained interval selection. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 302–313, 2012. `doi:10.1007/978-3-642-31594-7_26`.

25 Thomas Erlebach and Jirı Fiala. The maximum independent set problem in unit disk graphs, 2003.

26 Thomas Erlebach, Klaus Jansen, and Eike Seidel. Polynomial-time approximation schemes for geometric intersection graphs. *SIAM Journal on Computing*, 34(6):1302–1323, 2005.

27 Cristian Estan, George Varghese, and Mike Fisk. Bitmap algorithms for counting active flows on high speed links. In *Proceedings of the 3rd ACM SIGCOMM conference on Internet measurement*, pages 153–166. ACM, 2003.

28 Philippe Flajolet, Éric Fusy, Olivier Gandouet, and Frédéric Meunier. Hyperloglog: the analysis of a near-optimal cardinality estimation algorithm. In *Discrete Mathematics and Theoretical Computer Science*, pages 137–156. Discrete Mathematics and Theoretical Computer Science, 2007.

29 Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

30 Robert J Fowler, Michael S Paterson, and Steven L Tanimoto. Optimal packing and covering in the plane are np-complete. *Information processing letters*, 12(3):133–137, 1981.

31 Fănică Gavril. Algorithms for minimum coloring, maximum clique, minimum covering by cliques, and maximum independent set of a chordal graph. *SIAM Journal on Computing*, 1(2):180–187, 1972.

32 Phillip B Gibbons and Srikanta Tirthapura. Estimating simple functions on the union of data streams. In *Proceedings of the thirteenth annual ACM symposium on Parallel algorithms and architectures*, pages 281–291. ACM, 2001.

33 William K Hale. Frequency assignment: Theory and applications. *Proceedings of the IEEE*, 68(12):1497–1514, 1980.

34 Johan Håstad. Clique is hard to approximate within $n^{1\text{-epsilon}}$. In *37th Annual Symposium on Foundations of Computer Science, FOCS '96, Burlington, Vermont, USA, 14-16 October, 1996*, pages 627–636, 1996. `doi:10.1109/SFCS.1996.548522`.

35 Petr Hliněnỳ. Contact graphs of line segments are np-complete. *Discrete Mathematics*, 235(1-3):95–106, 2001.

36 Dorit S Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and vlsi. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

37 Hiroshi Imai and Takao Asano. Finding the connected components and a maximum clique of an intersection graph of rectangles in the plane. *Journal of algorithms*, 4(4):310–323, 1983.

38 Hossein Jowhari, Mert Sağlam, and Gábor Tardos. Tight bounds for lp samplers, finding duplicates in streams, and related problems. In *Proceedings of the thirtieth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 49–58. ACM, 2011.

39 Daniel M Kane, Jelani Nelson, and David P Woodruff. On the exact space complexity of sketching and streaming small norms. In *Proceedings of the twenty-first annual ACM-SIAM symposium on Discrete Algorithms*, pages 1161–1178. Society for Industrial and Applied Mathematics, 2010.

40 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 2010, June 6-11, 2010, Indianapolis, Indiana, USA*, pages 41–52, 2010. doi:10.1145/1807085.1807094.

41 Jon Kleinberg and Eva Tardos. *Algorithm design*. Pearson Education India, 2006.

42 Antoon WJ Kolen, Jan Karel Lenstra, Christos H Papadimitriou, and Frits CR Spieksma. Interval scheduling: A survey. *Naval Research Logistics (NRL)*, 54(5):530–543, 2007.

43 Christian Konrad. Maximum matching in turnstile streams. In *Algorithms-ESA 2015*, pages 840–852. Springer, 2015.

44 Ping Li and Cun-Hui Zhang. A new algorithm for compressed counting with applications in shannon entropy estimation in dynamic data. In *Proceedings of the 24th Annual Conference on Learning Theory*, pages 477–496, 2011.

45 Ewa Malesinska. *Graph theoretical models for frequency assignment problems*. Shaker, 1997.

46 Andrew McGregor. Graph stream algorithms: a survey. *ACM SIGMOD Record*, 43(1):9–20, 2014.

47 Peter Bro Miltersen, Noam Nisan, Shmuel Safra, and Avi Wigderson. On data structures and asymmetric communication complexity. *J. Comput. Syst. Sci.*, 57(1):37–49, 1998. doi:10.1006/jcss.1998.1577.

48 Kevin Mote. Fast point-feature label placement for dynamic visualizations. *Information Visualization*, 6(4):249–260, 2007.

49 Xiaoming Sun, David P Woodruff, Guang Yang, and Jialin Zhang. Querying a matrix through matrix-vector products. *arXiv preprint*, 2019. arXiv:1906.05736.

50 Gerhard J Woeginger. On-line scheduling of jobs with fixed start and end times. *Theoretical Computer Science*, 130(1):5–16, 1994.

A Arbitrary Length Intervals in Turnstile Streams

We now focus on estimating α and β for arbitrary-length intervals in turnstile streams. While we cannot obtain streaming algorithms in general, we show it is possible to estimate α and β when the maximum degree of the interval intersection graph or the maximum length of an interval arre bounded. In particular, we show an algorithm that achieves a $(1+\epsilon)$-approximation to α given that the maximum degree is upper bounded by $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. We also parameterize the problem with respect to the maximum length of an interval, W_{max} (assuming the minimum length is 1), and give an algorithm using $\text{poly}\left(W_{max}\frac{\log(n)}{\epsilon}\right)$ space.

A.1 Algorithms under Bounded Degree Assumptions

In light of the lower bound, we study the problem of estimating α for arbitrary-length intervals assuming the number of pair-wise intersections are bounded by $\kappa_{\max} = \text{poly}\left(\frac{\log(n)}{\epsilon}\right)$. In this section we show the following theorem:

▶ **Theorem 17.** *Let $\mathcal{P}$ be an turnstile stream of unit-weight arbitrary-length intervals with lengths polynomially bounded in n and let $\epsilon \in (0, 1/2)$. Let $\kappa_{\max} = poly\left(\frac{\log(n)}{\epsilon}\right)$ be the maximum number of pairwise intersections in $\mathcal{P}$. Then, there exists an algorithm that outputs an estimator Y such that the following guarantees hold:*

1. *$\frac{\alpha}{(1+\epsilon)} \leq Y \leq \alpha$ with probability at least $2/3$.*
2. *The total space used is $poly\left(\frac{\log(n)}{\epsilon}\right)$.*

Algorithm 3 Level Estimator.

Input: Given a turnstile stream $\mathcal{P}$ with unit weight arbitrary length intervals, where the length is polynomially bounded, $\epsilon > 0$ and $\delta > 0$, the algorithm outputs a $(1+\epsilon)$-approximation to α, assuming that $\kappa_{\max} = \text{poly}\left(\frac{log(n)}{\epsilon}\right)$.

1. Let $t = O\left(\frac{\log(n)}{\epsilon}\right)$ be the number of level-classes. Let $\Delta = \bigcup_{i\in[t]} \Delta_i$ be a randomly shifted Nested Grid, where Δ_i is a grid of side length $\frac{(1+\epsilon)^{i+1}}{\epsilon}$.
2. For $i \in [t]$, let $\mathcal{R}_i$ be the set of all r_i-Structures at level i, where a r_i-Structure is a subset of the Nested Grid, Δ, such that there exists an interval at the i^{th} level of the structure, there exist no intervals in the structure at any level $i' > i$ and all the intervals in the structure at levels $i' < i$ intersect the interval at the i^{th} level.
3. For all $i \in [t]$, using Algorithm 4, sample $\text{poly}\left(\frac{\log(n)}{\epsilon}\right)$ r_i-Structures from the set $\mathcal{R}_i$ to create a substream R_i^s. Note, this sampling is carried out with probability p_i defined below.
4. At the end of the stream, we recover R_i^s, for all $i \in [t]$. Let $Y_i = \frac{|\texttt{OPT}_{R_i^s}|}{p_i}$ (where p_i is the sampling probability for the i^{th} level), where $|\texttt{OPT}_{R_i^s}|$ can be computed using an offline algorithm.

Output: $Y = \sum_{i\in[t]} Y_i$.

Let W be the maximum length of the intervals in our input. We split our input into $t = O\left(\frac{\log(n)}{\epsilon}\right)$ length classes $\mathcal{W}_i$ such that for all $i \in [t]$, $\mathcal{W}_i = \{D_j \in \mathcal{P} | (1+\epsilon)^i \leq r_j \leq (1+\epsilon)^{i+1}\}$. Let $\mathcal{W}$ denote $\bigcup_{i\in[t]} \mathcal{W}_i$. We note that the partition here is over the input to the problem.

We can estimate the number of non-empty cells in each weight class up to a $(1 \pm \epsilon)$-factor by creating a substream for each one and running an ℓ_0 estimator on them. At the end of the stream, we can discard classes that are not within $\log(W)$ non-empty cells of each other. Therefore, we can assume the remaining classes have the same number of non-empty cells up to a $\log(W)$ factor.

We then make $O(\log(n))$ guesses for the number of non-empty cells for any fixed level and run our algorithm in parallel for each guess. Since there are t levels, this gives rise to an $O(t \log(n))$ factor blowup in space. At the end of the stream we know the correct value for each level via the ℓ_0 estimates. Let the number of non-empty cells at every level be denoted by X_i.

In contrast with our previous algorithm, we note that placing a grid on the input with side length 1 no longer suffices since our intervals may now lie in multiple cells. Therefore, we impose a nested grid over the input space:

▶ **Definition 18** (Nested Grid). *Given a partition $\mathcal{W}$, let grid Δ_i, corresponding to $\mathcal{W}_i \in \mathcal{W}$, be a set of cells over the input space with length $\frac{(1+\epsilon)^{i+1}}{\epsilon}$. Then a Nested Grid, denoted by Δ, is defined to be $\bigcup_{i\in[t]} \Delta_i$.*

We then randomly shift the nested grid such that at most an ϵ-fraction of intervals in the i^{th} length class lie within a distance $(1+\epsilon)^{i+1}$ of the i^{th} grid. Since this holds for all $\mathcal{W}_i$, and $\mathcal{W}_i$ are a partition of our input, we lose at most an ϵ-fraction of α. We then define the following object that enables us to obtain accurate estimates for each length class.

▶ **Definition 19** (r_i-Structure). *We define an r_i-Structure to be a subset of the Nested Grid, Δ, such that there exists an interval at the i^{th} level of the structure, there exist no intervals in the structure at any level $i' > i$ and all the intervals in the structure at levels $i' < i$ intersect the interval at the i^{th} level.*

Algorithm 4 Sampling r_i-Structures from $\mathcal{R}_i$.

Input: Given a turnstile stream $\mathcal{P}$ with unit weight arbitrary length intervals, with the length being polynomially bounded, $\epsilon > 0$ and $\delta > 0$, the sampling procedure creates a poly $\left(\frac{\log(n)}{\epsilon}\right)$ size sample of the set $\mathcal{R}_i$.

1. Let Δ_i be the i^{th} level of a randomly shifted *Nested Grid* Δ. Let $\mathcal{R}_i$ be the set of r_i-Structures where the topmost cells lie in Δ_i. Let X_i be the correct guess for the number of non-empty cells in Δ_i up to a constant.
2. Agnostically sample cells from Δ_i with probability $p_i = \max\left(\text{poly}\left(\frac{\log(n)}{\epsilon}\right)\frac{1}{X_i}, 1\right)$. Let S_i be the corresponding substream created.
3. For each cell $c \in \mathcal{S}_i$, let r_i^c be a structure (as defined in 5) with c at the topmost level. Run κ_{max}-*Sparse Recovery* on substream $\mathcal{S}_i$.
4. At the end of the stream, verify that r_i^c is a valid r_i-Structure. Let R_i^s be the set of all such structures.
5. If $X_i > \frac{\epsilon \sum_{i\in[t]} X_i}{t}$, keep R_i^s, else discard it.

Output: $\bigcup_{i\in[t]} R_i^s$.

Let $\mathcal{R}_i$ denote the set of all r_i-Structures at level i. Observe that, taking the union over $i \in [t]$ of $\mathcal{R}_i$ gives a partition of the input. Therefore, estimating $|\texttt{OPT}_{\mathcal{R}_i}|$ separately and summing up the estimates is a good estimator for α.

Similar to the algorithm in Section 5 a key tool we use is k-Sparse Recovery. Intuitively, we subsample poly$\left(\frac{\log(n)}{\epsilon}\right)$ r_i-Structures from the set $\mathcal{R}_i$ to create a substream $\mathcal{R}_i^s$ and run a $\kappa_{\max}$-Sparse Recovery Algorithm on each substream. At the end of the stream, we get an estimate of $|\texttt{OPT}_{\mathcal{R}_i}|$ that concentrates. We then add up the estimates across all the levels to form our overall estimate. We formally describe the Level Estimator Algorithm in Algorithm 3, assuming we are given access to a black-box sampling algorithm for sampling an r_i-Structure. We describe how to sample r_i-Structures in *turnstile streams* in Algorithm 4.

Next, we consider the problem of estimating α for arbitrary-length intervals assuming that the space available is at most poly$\left(\frac{W_{max}\log(n)}{\epsilon}\right)$, where W_{max} is an upper bound on the ratio of the max to the min length of an interval. We note that this regime is interesting when W_{max} is sublinear in n. We obtain the following result:

▶ **Theorem 20.** *Let $\mathcal{P}$ be an turnstile stream of unit-weight arbitrary-length intervals s.t. the length is polynomially bounded in n and let $\epsilon \in (0, 1/2)$. Let $\mathcal{W}_{max}$ be an upper bound on the ratio of the max to the min length of intervals in $\mathcal{P}$. Then, there exists an algorithm that outputs an estimator Y s.t. the following guarantees hold:*

1. $\frac{\alpha}{(2+\epsilon)} \leq Y \leq \alpha$ *with probability at least* $2/3$.
2. *The total space used is poly* $\left(\frac{\mathcal{W}_{max}\log(n)}{\epsilon}\right)$.

B Unit Radius Disks in Turnstile Streams

In this section, we state our main result for approximating α and β for unit-radius disks in $\mathbb{R}^2$ that are received in a turnstile stream. Given space constraints, we defer the exposition to the full version.

The main algorithmic result we prove is the following:

▶ **Theorem 21.** *Let $\mathcal{P}$ be a sequence of unit-radius disks that are received as a turnstile stream and let $\epsilon \in (0, 1/2)$. Then, there exists an algorithm that outputs an estimator Y such that with probability at least $9/10$, $\left(\frac{\pi}{8\sqrt{3}} + \epsilon\right)\beta \leq Y \leq \beta$ where α is the cardinality of the largest independent set in $\mathcal{P}$. Further, the total space used is $O\left(\text{poly}\left(\frac{\log n}{\epsilon}\right)\right)$.*

C Insertion-Only Streams

In this section, we state our results for estimating the maximum weighted independent set of intervals in insertion-only streams. Recall, [10] show that $\left(\frac{3}{2} + \epsilon\right)$ is tight for the unweighted case in insertion-only streams. We also show a lower bound for estimating the maximum independent set of disks in insertion-only streams. The lower bound for intervals in [10] shows that $\left(\frac{3}{2} - \epsilon\right)$-approximation requires $\Omega(n)$ space and this naturally extends to disks. We improve this to $2 - \epsilon$, implying a strict separation between intervals and disks for insertion-only streams. Note, this is not yet known to be the case for turnstile streams.

Our theorem for weighted MIS of unit interval in insertion-only streams is as follows:

▶ **Theorem 22.** *Let P be an insertion-only stream of weighted unit intervals s.t. the weights are polynomially bounded in n and let $\epsilon \in (0, 1/2)$. Then, there exists an algorithm that outputs an estimator Y s.t. with probability at least $9/10$ the following guarantees hold:*

1. $\frac{2\beta}{3+\epsilon} \leq Y \leq \beta$.
2. *The total space used is poly $\left(\frac{\log(n)}{\epsilon}\right)$ bits.*

Next, we describe a lower bound for estimating α for unit disks in insertion-only streams via a reduction from the communication complexity of the `Indexing` problem, which we use as the starting point. We consider the one-way communication model between two players Alice and Bob and each player has access to private randomness. The randomized communication complexity of `Indexing` is well understood in the two-player one-way communication model.

▶ **Definition 23** (`Indexing`). *Let $\mathtt{I}_{n,j}$ denote the communication problem where Alice receives as input a bit vector $x \in \{0,1\}^n$ and Bob receives an index $j \in [n]$. The objective is for Bob to output x_j under the one-round one-way communication model with error probability at most $1/3$.*

▶ **Theorem 24** (Communication Complexity of $\mathtt{I}_{n,j}$). *The randomized one-round one-way communication complexity of $\mathtt{I}_{n,j}$ with error probability at most $1/3$ is $\Omega(n)$.*

We begin with considering the stream of disks $\mathcal{P}$. Let `Alg` be a one-pass insertion-only streaming algorithm that estimates the cardinality of the maximum independent set denoted by α. We then show that `Alg` can be used as a subroutine to solve the communication problem $\mathtt{I}_{n,j}$. Therefore, a lower bound on the communication complexity in turn implies a lower bound on the space complexity of `Alg`. Formally,

▶ **Theorem 25.** *Given a stream of disks $\mathcal{P}$, any randomized one-pass insertion-only streaming algorithm* `Alg` *which approximates α to within a $(2-\epsilon)$-factor, for any $\epsilon > 0$, with error at most $1/3$, requires $\Omega(n)$ space.*

Proof. We show that any such insertion-only streaming algorithm `Alg` can be used to construct a randomized protocol Π to solve the communication problem. Given her input x, Alice constructs a stream of unit disks and runs `Alg` on the stream. Consider the unit circle around the origin. Divide the half-circle above the x-axis into n equally spaced points, denoted by vectors $p_1, p_2, \ldots, p_n$. For $i \in [n]$, if $x_i = 0$, Alice streams a unit disk centered at p_i. If $x_i = 1$, Alice streams a unit disk centered at $-p_i$. After streaming n disks, Alice communicates the memory state of `Alg` to Bob. Bob uses the message received from Alice as the initial state of the algorithm and continues the stream. Recall, Bob's input only consists of a single index j. Therefore, Bob inserts a unit disk centered at $(1 + 1/n^2)p_j$.

We first observe that all disks inserted by Alice pairwise intersect. Since all her unit radius disks are centered on the unit circle around the origin, the distance between their center and the origin is 1. Since all the disks contain the origin, they pairwise intersect. Now, let us consider the case where $x_j = 0$. Recall, in this case, Alice inserts the disk centered p_j and Bob inserts the disk centered at $(1 + 1/n^2)p_j$. The distance between their centers is $1/n^2$ and they clearly intersect. Let us now consider the other disks inserted by Alice, centered at points p_i for $i \neq j$. The distance between their centers is

$$\begin{aligned} ||p_i - (1+1/n^2)p_j||_2^2 &= ||p_i||_2^2 + (1+1/n^2)^2||p_j||_2^2 \pm 2(1+1/n^2)\langle p_i, p_j\rangle \\ &\leq 1 + (1+3/n^2) \pm 2(1+1/n^2)\langle p_i, p_j\rangle \end{aligned} \tag{C.1}$$

where the last inequality follows from $(1+1/n^2)^2 = 1+1/n^4+2/n^2 \leq 1+3/n^2$ for sufficiently large n. Since $i \neq j$, $\langle p_i, p_j\rangle \leq 1 - \Theta(1/n)$. Note, $(1+1/n^2)(1-\Theta(1/n)) \leq 1 - \Theta(n)$ for sufficiently large n. Substituting this above, we get

$$\begin{aligned} ||p_i - (1+1/n^2)p_j||_2^2 &\leq 1 + (1+3/n^2) \pm 2(1+1/n^2)(1-\Theta(1/n)) \\ &\leq 2 + 3/n^2 \pm 2(1-\Theta(1/n)) \\ &\leq 4 - \Theta(1/n) \end{aligned} \tag{C.2}$$

where the last inequality follows from $\Theta(1/n) \geq 3/n^2$ for sufficiently large n. Therefore, the squared distance between the centers is strictly less 4 and the disks do intersect. As a consequence, all disks pairwise intersect and $\alpha = 1$.

Let us now consider the case where $x_j = 1$. Recall Alice inserts a disk centered at $-p_j$ and Bob inserts a disk centered at $(1 + 1/n^2)p_j$. The distance between the centers is $(2 + 1/n^2)$, therefore the two disks do not intersect. Then, α is at least 2. We observe that any $(2-\epsilon)$-approximate algorithm `Alg` can distinguish between these two cases because in the first case `Alg` outputs *at most* 1 and in the second case `Alg` outputs *at least* $1 + \epsilon$. Therefore it is a valid protocol for solving $\mathtt{I}_{n,j}$. If `Alg` has error at most $1/3$, the protocol has error at most $1/3$. By Theorem 24, any such protocol requires $\Omega(n)$ communication and in turn `Alg` requires $\Omega(n)$ space. ◀